Different Learning Methods for Machine-Generated Text Detection

Ng Man Tik
1155158302
1155158302@link.cuhk.edu.hk

Abstract

This work investigates various methodologies, including Support Vector Machines (SVM) with RBF kernels, Bidirectional Long Short-Term Memory (BiLSTM) networks with attention mechanisms, and the Mistral model, for classifying machine-generated text. We evaluate these models on the M4 dataset and MGTBench for out-of-distribution data, as well as on mixed data within MixSet. Our results indicate that SVMs equipped with RBF kernels and BiLSTMs augmented with attention mechanisms significantly outperform other models in their respective categories. Notably, they achieve perfect accuracy on unseen GPT-4 generated data. Additionally, we find that the choice of data mixing method crucially impacts the effectiveness of the detectors, with the humanization approach posing the greatest challenge.

1 Introduction

The advent of generative models, particularly ChatGPT and GPT-4, marks a significant evolution in artificial intelligence, profoundly impacting various fields such as academic writing, story generation, and software development Lee et al. [2023], Pagnoni et al. [2022], Mirsky et al. [2022], Stokel-Walker [2022], Kasneci et al. [2023]. The capabilities of Large Language Models (LLMs) have evolved to produce text nearly indistinguishable from human writing, as evidenced by recent studies Chowdhery et al. [2022]. However, this technological leap brings forth new challenges, notably the difficulty in distinguishing between AI-generated and human-authored texts. This ambiguity raises concerns regarding information quality—given LLMs' dependency on potentially outdated or biased datasets—and the potential for misuse in areas like fake news dissemination and academic dishonesty Yuan et al. [2022], Becker et al. [2023], Zheng et al. [2023].

Current research efforts have focused on developing methods to detect machine-generated content, typically through fine-tuning existing language models with extensive datasets. However, these approaches often overlook the nuanced reality where texts are neither purely machine-generated nor entirely human-written, failing to reflect the complex interactions between AI and human input in real-world applications.

This project advances the detection of AI-generated texts by integrating Long Short-Term Memory (LSTM) networks, Convolutional Neural Networks (CNNs), and Attention mechanisms. This approach aims to discern nuanced differences between AI-generated and human texts effectively, improving accuracy across various text sources and styles, particularly those blending AI and human inputs. The significance of this research extends to addressing ethical standards, copyright laws, and transparency in digital content creation. By enhancing methods to identify AI-generated content, this project contributes to discussions on AI's role in content authenticity and human creativity in the digital age. Through these efforts, we provide tools and insights to responsibly navigate the evolving landscape of AI-generated content.

2 Related Work

2.1 Text Classification in Natural Language Processing

Text classification serves as a cornerstone in the field of Natural Language Processing (NLP), essential for tasks ranging from sentiment analysis to fake news detection. Traditionally, this field has relied on machine learning techniques like Naïve Bayes, Decision Trees, and Support Vector Machines (SVMs), utilizing feature extraction methods such as bag-of-words or TF-IDF Li et al. [2015], Gurkhe et al. [2014], Pang et al. [2002]. However, the emergence of Large Language Models (LLMs) like ChatGPT has shifted the paradigm, making the detection of machine-generated text increasingly complex due to their advanced human-like writing styles. This transformation underscores a critical challenge: distinguishing between human and LLM-generated texts, which have become remarkably similar, blurring the lines of authorship Guo et al. [2023], Ma et al. [2023], Muñoz-Ortiz et al. [2023].

2.2 Evolution of Models - LSTM, CNN, and Hybrid Approaches

In the domain of text classification, the introduction of deep neural networks such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) has initiated a paradigm shift from traditional feature extraction methods to more dynamic and intuitive analysis processes. Unlike earlier methods that relied heavily on manual feature engineering, CNNs and RNNs have paved the way for automatic pattern recognition in text, significantly enhancing the efficiency and effectiveness of text classification.

Specifically, CNNs excel in identifying local textual features, providing detailed insights into the structure and composition of the text, while RNNs, especially Long Short-Term Memory networks (LSTMs), are adept at understanding sequence dependencies, capturing the temporal and contextual nuances of written language Giorgi et al..

The convergence of CNNs and RNNs has led to the development of hybrid models that combine the strengths of both architectures, offering a more comprehensive approach to text analysis. This integrated approach significantly improves feature extraction within text sequences, leading to enhanced model accuracy and interpretability Wu and Deng [2022], Wang et al. [2016], Deng et al. [2021], Liu and Guo [2019], Cheng et al. [2020], Bahdanau et al. [2014], Du et al. [2017], SiChen [2019]. Innovations in this area, such as those by G. Wang et al., have introduced methods for embedding label sets into vector spaces, facilitating more effective computation and analysis of text data Wang et al. [2018]. Additionally, the integration of self-attention and label-embedding techniques, as demonstrated by Dong, Y et al. Dong et al. [2019] and Y. Xiao et al. Xiao et al. [2021], further enriches model capabilities, enabling more focused and relevant analyses of textual content.

One notable application of this combined approach is the utilization of CNN and Attention mechanisms as encoders with BiLSTM decoders. This configuration has been effectively applied in scenarios such as stock prediction, showcasing the model's ability to interpret complex sequential data accurately. Similarly, the amalgamation of CNNs and BiLSTMs with interactive Attention mechanisms has proven beneficial in critical areas like fake news detection, underlining the model's proficiency in identifying subtle semantic nuances and patterns within texts.In our project, we intend to further refine this combined model architecture by integrating CNNs with BiLSTMs and embedding strategic attention layers to enhance feature extraction and interpretation. This refined approach is specifically aimed at improving the model's ability to differentiate between texts generated by humans and those produced by LLMs, a task growing ever more challenging with the advancing capabilities of modern language models. By fine-tuning the interaction between these model components and adjusting their configurations, we anticipate not only higher accuracy in text classification but also a deeper insight into the distinguishing characteristics of human versus LLM-generated texts.

2.3 Fine-tuning LLMs for Text Classification

The advent of transformer architectures has introduced a new frontier in NLP, with models like BERT, Roberta, and XLNet setting new benchmarks in text understanding and classification Qiu et al. [2020], Devlin et al. [2019], Liu et al. [2019], Yang et al. [2019], Wang et al. [2019]. These models' finetuning, particularly for tasks like distinguishing LLMgenerated texts, has shown promising results. However, the computational demand of these models poses a significant barrier for individuals with limited resources. Our work seeks to address this by employing a smaller, more efficient model, Mistral-7B, leveraging techniques such as LoRA to enable fine-tuning with reduced resource requirements Jiang et al. [2023].

2.4 Addressing Potential Attacks on Text Classification Models

Despite achieving high performance in identifying machine-generated texts, models remain susceptible to various adversarial attacks that could significantly impair their effectiveness. Recent research highlights that even high-performing models can falter when confronted with specific, subtly altered inputs. For instance, the application of a lightweight paraphrase model to alter the wording and semantic distribution of machine-generated texts has demonstrated potential in undermining zero-shot detection capabilities Sadasivan et al. [2023], Orenstrakh et al. [2023]. This reveals the models' vulnerability to nuanced changes that preserve meaning while altering textual structures.

Further complicating the landscape, Shi et al. Shi et al. [2023] and He et al. He et al. [2023a] have documented the efficacy of permutation strategies in deceiving text detection systems. Techniques such as content cutoff Shen et al. [2020], sequence shuffling Lee et al. [2020], token mutation Liang et al. [2023], and strategic word swapping Shi and Huang [2020] pose significant challenges, indicating that these methods can effectively mask the machine-generated nature of texts, thereby evading detection by otherwise robust models.

In response to these challenges, our project plans to leverage the MixSet dataset Chen [2024], renowned for its incorporation of texts that blend human and machine elements. This dataset serves as a critical resource for simulating real-world applications, where texts often exhibit characteristics of both human and AI contributions. By employing this dataset, we aim to evaluate and enhance the resilience of our models against a range of adversarial tactics. Specifically, we will investigate the model's performance against paraphrased outputs—a common form of attack aiming to 'humanize' machine-generated content. This approach will not only test the models' detection capabilities under manipulated conditions but also contribute to the ongoing discourse on securing AI-driven text analysis tools against emerging threats.

3 Data

This study leverages three primary datasets, each offering unique insights and challenges relevant to distinguishing between AI-generated and human-written texts. Below is a detailed exploration of these datasets:

M4 Dataset The M4 dataset Wang et al. [2023] is designed to evaluate machine-generated data across multiple generators, domains, and languages. Given the complexity and extensive workload required to handle this dataset, this study focuses only on the multi-domain and multi-generator aspects. The selected domains and generators included in this study are illustrated in the following figure:

Source/ Domain	**Language**	**Total Human**	**Parallel Data**						
			Human	**Davinci003**	**ChatGPT**	**Cohere**	**Dolly-v2**	**BLOOMz**	**Total**
Wikipedia	English	6,458,670	3,000	3,000	2,995	2,336	2,702	3,000	17,033
Reddit ELI5	English	558,669	3,000	3,000	3,000	3,000	3,000	3,000	18,000
WikiHow	English	31,102	3,000	3,000	3,000	3,000	3,000	3,000	18,000
PeerRead	English	5,798	5,798	2,344	2,344	2,344	2,344	2,344	17,518
arXiv abstract	English	2,219,423	3,000	3,000	3,000	3,000	3,000	3,000	18,000
Baike/Web QA	Chinese	113,313	3,000	3,000	3,000	–	–	–	9,000
RuATD	Russian	75,291	3,000	3,000	3,000	–	–	–	9,000
Urdu-news	Urdu	107,881	3,000	–	3,000	–	–	–	9,000
id_newspapers_2018	Indonesian	499,164	3,000	–	3,000	–	–	–	6,000
Arabic-Wikipedia	Arabic	1,209,042	3,000	–	3,000	–	–	–	6,000
True & Fake News	Bulgarian	94,000	3,000	3,000	3,000	–	–	–	9,000
Total			35,798	23,344	32,339	13,680	14,046	14,344	133,551

Figure 1: Selected domains and generators in the M4 dataset

A significant challenge with the M4 dataset is the data imbalance between human-generated and machine-generated texts, typically in a 1:3 ratio. To address this, oversampling of the human-generated data is performed by replicating the dataset three times, ensuring a balanced distribution for effective model training and evaluation:

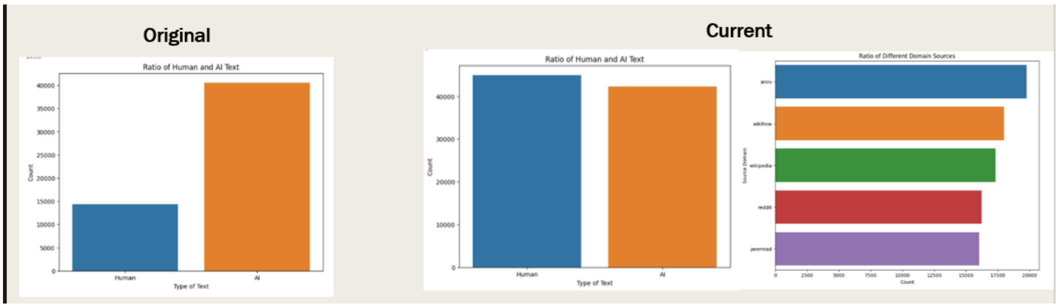


Figure 2: Comparative data distribution using the original and new oversampling approaches

The dataset comprises a total of 85,000 entries, spanning training, validation, and testing datasets.

MGTBench MGTBench He et al. [2023b], similar to the M4 dataset, evaluates the performance of models on machine-generated text but focuses primarily on out-of-distribution (OOD) data. It also incorporates multiple domains and generators, providing a comprehensive test environment. The components included in the MGTBench dataset are detailed below:

This dataset is particularly used to test the resilience of models against texts that were generated through sophisticated methods, potentially simulating advanced adversarial scenarios.

Table 1: The selected domain and Generator for MGTBench (The italic ones are out-of-distribution)

Domain Used	Generator Used
News	Human
Essay	ChatGPT
Story	*GPT4*

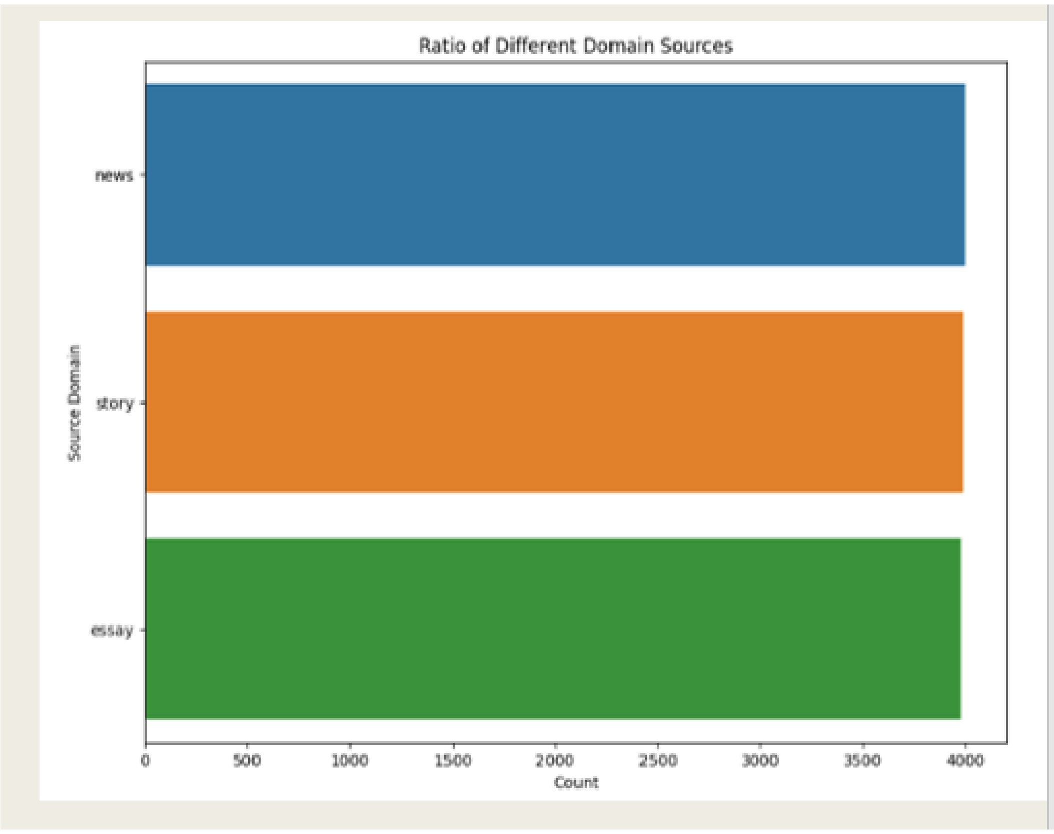


Figure 3: MGTBench Domain Distribution

MixSet MixSet is a dataset that integrates machine-generated and human-written textual features. It comprises four distinct methods of data mixing:

- **Polish:** The language model refines content at the sentence or word level to enhance clarity and style.
- **Rewrite:** The model extracts essential information from the content and rearticulates it entirely.
- **Complete:** After reviewing the initial third of the data, the model generates the remaining two-thirds to complete the text.
- **Humanize:** This method involves embedding human-like textual features into the original machine-generated data, making it appear more naturally written.

The source of the original data for the Polish, Rewrite, and Complete methods is human-written, whereas for the Humanize method, it is machine-generated.

This dataset's composition necessitates a unique classification goal to effectively evaluate model performance. The fundamental challenge is determining "whether the model can be deceived by the language model into misclassifying the data's source." Based on this, specific criteria for correction are established to calculate accuracy and other metrics, as shown 5.

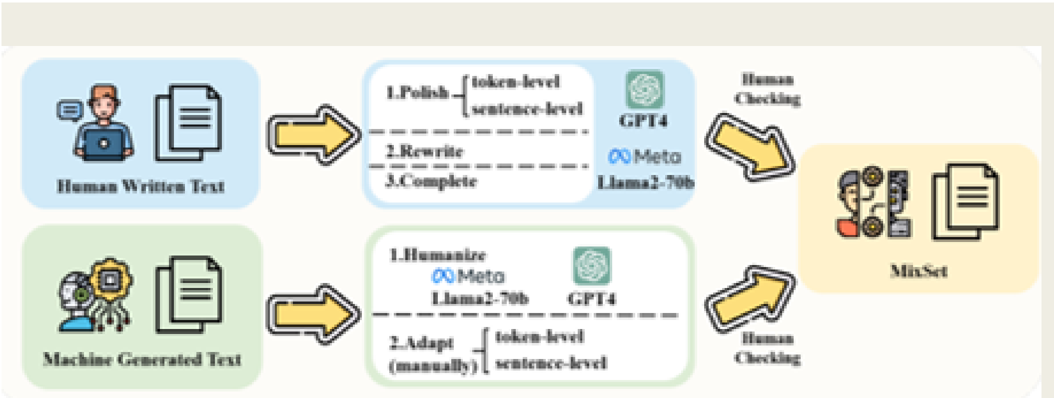


Figure 4: Overview of MixSet

Method	Correct Criteria
Polish	HGT / 0
Rewrite	HGT / 0
Complete	HGT / 0
Humanize	MGT / 1

Figure 5: Label Criteria for MixSet

For practical purposes, only data mixed using GPT-4 are selected for testing to maintain consistency and control in experiment conditions. Given the limited size of the original dataset, the test dataset comprises approximately 750 entries.

4 Approach

4.1 Dataset preprocessing

Before loading the datasets for training and testing process, prepare the data from the dataset first. It first needs to remove all the dataset where its language is not English in the dataset. Then, it needs to read different json files and then copy the machine-text or human-text with manually labeling as 0 if it is human-text and 1 if machine-text. After that, we need to clean the data by removing rows with missing values in the 'text' column to ensure data quality. Finally, the result data neeed to be normalized and used for further process.

4.2 Text Processing

For the text processing, we applied standard natural language processing techniques taught in lecture because the feature engineering part will be performed by CNN and Attention in the model. We first tokenized the textual content using a basic English tokenizer for splitting text into words and tokens while removing all the punctuation and stop words then construct a vocabulary with 10000 most frequent tokens to reduce computational complexity and memory requirements. Finally, I encoded the text by replacing the token to index for processing by neural networks with padding and truncation to standardizing the sequences with 200tokens only.

4.3 SVM

4.3.1 General Model Architecture

To task the performance of those traditional machine-learning method for classification of LLM-generated text with human text, I incorporated the SVM for text classification.

Before deploying the model, the preprocessed the data will further employs the TFIDF to converts the text data into a matrix of TFIDF features to enabling the SVM to know the textual features. The TFIDF vectors is set with the maximum length of 1000. After that, the radius basis function will be act as the kernel of SVM and do the classification.

4.3.2 Comparable Baselines

The model will be comparable with the same SVMs, but with different kernels which are linear, sigmoid and polynomic to understand how well each variant perform across the same text sources.

4.4 BiLSTM-Attention series models

The following sections will introduce all the curcial elements for that series of model and showcase the key different in their architectures.

4.4.1 Embedding Layer

The first layer is the embedding layer, after loading the preprocessed text, it will map each token to a high-dimensional vector using one-hot encoding to put them into dense representations to capture semantic properties such similar vectors for later layers to understand the content.

4.4.2 1D Convolution Layer

After converting to the word vectors, it will be applied into 1D CNN model to extract a representative and effective feature by performing a one-dimensional convolution operation with various filters in different sizes. Using various filters over the sequence data, the 1D CNN can capture hierarchical features inside the long sequence and passes the filtered information to the next layer. By applying more than one convolutions layers, the 1D CNN model deepens the feature extraction process. Thus a higher level of features makes the prediction task more robust and discriminative,

Assume the input from the embedding layer is with dimension d, those vectors will form an input matrix with dimensions corresponding to the sequence length and vector size which is L x d. Then the matrix can be processed by the multi-channel convolution layer that employs kernels of varying size in 2, 3, and 4 words to produce different feature map lengths in order to capture different local textual features. Those kernels will focus on different n-gram combinations while the global max pooling reduces the feature map into condensed representation. Then the Rcetified Linear Unit (ReLU) is used as the activation function for introducing the non-linearity into the model as well.

4.4.3 Attention mechanism

In general, for classifying the content if it is machine-generated or not, not all the parts of that content contribute equally to decide the final prediction. Some must be more important than others. In view of this, the utilization of the attention mechanism is performed to emphasize the most important parameters during prediction. In this series of models, the attention mechanism can be performed in either two places: After 1D CNN and before Bi-LSTM and after Bi-LSTM, we will call it Pre-LSTM attention and Post-LSTM Attention.

Pre-LSTM Attention After generating the feature maps using CNN layers, attention mechanism is used to aim to weigh the importance of different n-gram features extracted by the CNN layers before they are processed by the LSTM which can focus on more relevant features extracted from the convolutional layers. It does so by computing a weight sum of the features based on the attention weights, resulting in an attended feature vector which represents a focused summary of the most relevant features.

Post-LSTM Attention The Post-LSTM attention is applied to focus on local text features before sequence processing, the Post-LSTM Attention assesses the importance of different parts of the text after considering its full context. This attention step assigns weights to each position in the BiLSTM's output sequence, identifying which parts are most relevant for the classification decision. Using it can emphasize the most informative parts of the texts given from the output of BiLSTM layer to make the final classification decision. The result will finally input to the fully connected layer to give the binary classification of human (0) or machine-text (1).

4.4.4 Bi-direction LSTM

The Bi-directional LSTM network is a two-way stacked LSTM network with forward and backward LSTM features. This layer can be able to capture the long-term dependencies within sequence data with the additional information the feature sequences. The layer is using bi-directional version of LSTM to capture the context from both sides due to the fact that the meaning or choice of wording should be depend form both sides not just words before it. It can thus offer a more complete understanding of each word within its surrounding context.

4.4.5 Comparing Baselines

There are the following models used in the experiment, Bi-LSTM, Attention-Bi-LSTM, CNNBiLSTM, CNNBiLSTM-BiAttention and CNNBiLSTMAttention. The following table is the comparism between them:

- Ordinary version of BiLSTM: Using BiLSTM directly in extracting the sequential dependencies of the sequences for classification - Attention with BiLSTM: Attention is appended after BiLSTM for further focusing the important features in the result produced in BiLSTM - CNN with BiLSTM: CNN is performed in feature extraction before doing the classification with BiLSTM - CNNBiLSTM-Attention: A similar approach to the proposed model but removing the attention layer between CNN and BiLSTM to test the effectiveness of that layer - CNNBiLSTM-BiAttention: The model with all the layers including the pre-LSTM Attention and post-LSTM Attention.

4.5 Mistral 7B

The Mistral 7B model is another key component of this study. According to the model's foundational paper, Mistral 7B significantly outperforms the popular Llama 2 13B across all benchmarks, and even surpasses Llama 34B on many benchmarks. This demonstrates that smaller language models can match the capabilities of larger ones when optimized correctly Jiang et al. [2023]. For this project, we utilize the Mistral-7B variant enhanced with Block-wise Model-update Filtering and Bit-centering (BNB), which boosts model efficiency and reduces memory demands. Additionally, we employ a

Table 2: Comparison of Model Architectures

Layer (Specification)	BiLSTM	Att-BiLSTM	CNN-BiLSTM	CNNBiLSTM-Att	CNNBiLSTM-DouAtt
Embedding	vocab_size dim=128	vocab_size dim=128	vocab_size dim=128	vocab_size dim=128	vocab_size dim=128
CNN	-	-	filters=100 kernels=[2,3,4]	filters=100 kernels=[2,3,4]	filters=100 kernels=[2,3,4]
Pre-LSTM Att	-	-	-	-	heads=4 depth=per_head
Bi-LSTM	hidden=256 layers=2	hidden=256 layers=2	hidden=256 layers=2	hidden=256 layers=2	hidden=256 layers=2
Post-LSTM Att	-	heads=4 depth=per_head	-	heads=4 depth=per_head	heads=4 depth=per_head
Dense Output	classes=1	classes=1	classes=1	classes=1	classes=1

quantized 4-bit version of the model Face [2024a], facilitating training on T4 GPUs by minimizing the model's size.

In the implementation phase, the 'FastLanguageModel' from the UnSLoth library AI [2024] is used to download Mistral-7B and set the maximum sequence length to 2048 tokens. Furthermore, LoRA technology is applied to train only 4% of the model's parameters, utilizing techniques such as gradient accumulation and precision training to enhance training efficiency. Unlike the standard natural language processing approaches used with the SVM and LSTM models, this phase involves Supervised Fine-Tuning. Here, text data and their corresponding labels are formatted into prompts suitable for retraining the model on the machine-text classification task. Training is conducted using the PEFT technique Face [2024b] combined with the SFT Trainer Face [2024c], optimizing the model's performance in text classification.

5 Experiment

5.1 Text Processing

Text processing was performed using standard natural language processing (NLP) techniques. Initially, texts were tokenized using a basic English tokenizer that splits text into words and tokens, while removing all punctuation and stopwords. This process helped in constructing a vocabulary of the 10,000 most frequent tokens, aimed at reducing computational complexity and memory requirements. Subsequently, texts were encoded by replacing each token with its corresponding index, facilitating neural network processing. All texts were then standardized to sequences of 200 tokens through padding and truncation.

5.2 Training Augmentation

Following text preprocessing, we proceeded to the training phase for each model.

BiLSTM Series For the BiLSTM series, we utilized TF-IDF features to enable the SVM to recognize textual characteristics effectively. The TF-IDF vectors were configured with a maximum length of 1000. Subsequently, a radial basis function kernel was employed in the SVM for classification purposes.

Training for all models was uniformly conducted over 8 epochs to ensure fairness and consistency across evaluations.

Mistral7B Here are the setting for each part when training with Mistral-7B model:

```
model, tokenizer = FastLanguageModel.from_pretrained(
    model_name = "unsloth/mistral-7b-bnb-4bit", # Choose ANY! eg teknium/OpenHermes-2.5-Mistral-7B
    max_seq_length = max_seq_length,
    dtype = dtype,
    load_in_4bit = load_in_4bit,
    # token = "hf_...", # use one if using gated models like meta-llama/Llama-2-7b-hf
)
```

Figure 6: Enter Caption

```
model = FastLanguageModel.get_peft_model(
    model,
    r = 8, # Choose any number > 0 ! Suggested 8, 16, 32, 64, 128
    target_modules = ["q_proj", "k_proj", "v_proj", "o_proj",
                      "gate_proj", "up_proj", "down_proj",],
    lora_alpha = 32,
    lora_dropout = 0, # Supports any, but = 0 is optimized
    bias = 'lora_only',     # Supports any, but = "none" is optimized
    use_gradient_checkpointing = True,
    random_state = 3407,
    use_rslora = True,  # We support rank stabilized LoRA
    loftq_config = None, # And LoftQ
)
```

Figure 7: Lora and Peft setting

5.2.1 Loss Function

Given the inherent data imbalance in the M4 dataset, which typically includes one human-written text alongside several machine-generated texts from various models on a specific topic, addressing this imbalance was crucial. Although oversampling of human-generated data was implemented to mitigate this issue, it introduced a potential bias from the generator's perspective, possibly skewing the models towards the majority class.

To counteract this, we opted to use the Focal Loss function Lin et al. [2017] instead of the traditional binary classification loss. Focal Loss is designed to adjust the contribution of each example to the loss based on the classification error, emphasizing harder-to-classify examples and diminishing the

```python
from trl import SFTTrainer
from transformers import TrainingArguments

trainer = SFTTrainer(
        model = model,
        tokenizer = tokenizer,
        train_dataset = train_data,
        eval_dataset = val_data,
        dataset_text_field = "formatted_text",
        max_seq_length = max_seq_length,
        args = TrainingArguments(
                output_dir = "outputs",
                per_device_train_batch_size = 8,
                gradient_accumulation_steps = 8,
                warmup_steps = 5,
                max_steps = 60,
                learning_rate = 2e-4,
                fp16 = not torch.cuda.is_bf16_supported(),
                bf16 = torch.cuda.is_bf16_supported(),
                logging_steps = 1,
                optim = "adamw_8bit",
                weight_decay = 0.01,
                lr_scheduler_type = "linear",
                seed = 3407
        ),
)
```

Figure 8: SFTTrainer Setting

impact of well-classified instances. This approach, an extension of the standard Cross-Entropy Loss, enhances the models' sensitivity to minority classes and promotes a more balanced performance across different classes.

5.3 Training Log

This subsection details the training logs for SVM and BiLSTM series models, highlighting the computational efforts and key metrics observed during the training process.

5.3.1 SVM

Training the SVM model primarily involved utilizing built-in functions for model training and evaluation. There is no traditional "training log" for SVM as it directly returns the analysis results after the training session. However, the computational time required is noteworthy; training and evaluating a single kernel took approximately 4 hours. By reducing the size of the training dataset, this duration was decreased to about 1.5 hours, demonstrating a significant dependency of training time on dataset size.

5.3.2 BiLSTM series

The training log for the BiLSTM series models includes details on training and validation losses, as well as accuracies, providing insights into the models' performance through the training epochs. Below are the summarized logs for each model:

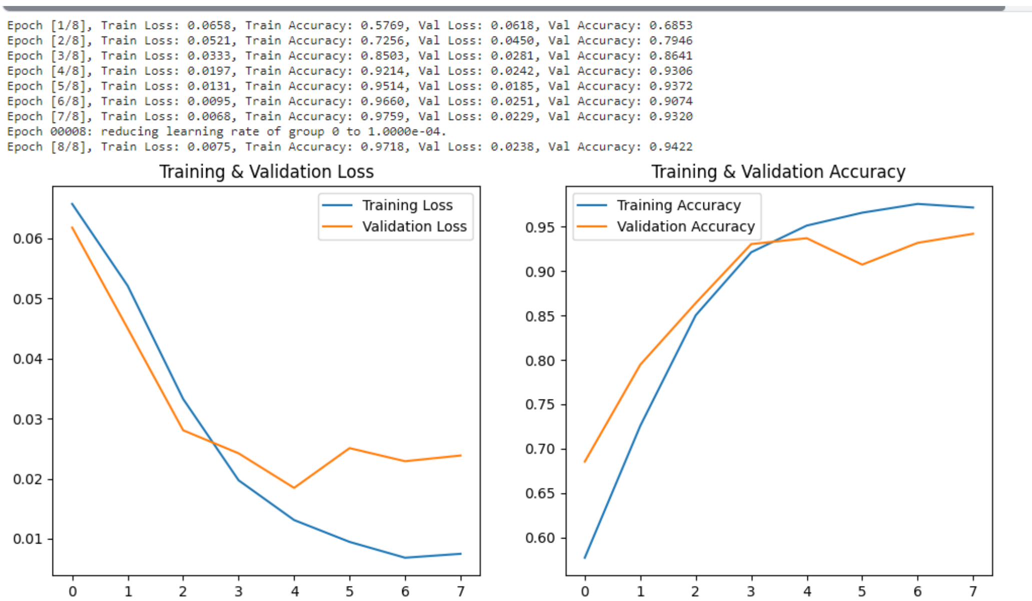


Figure 9: Training Log for BiLSTM

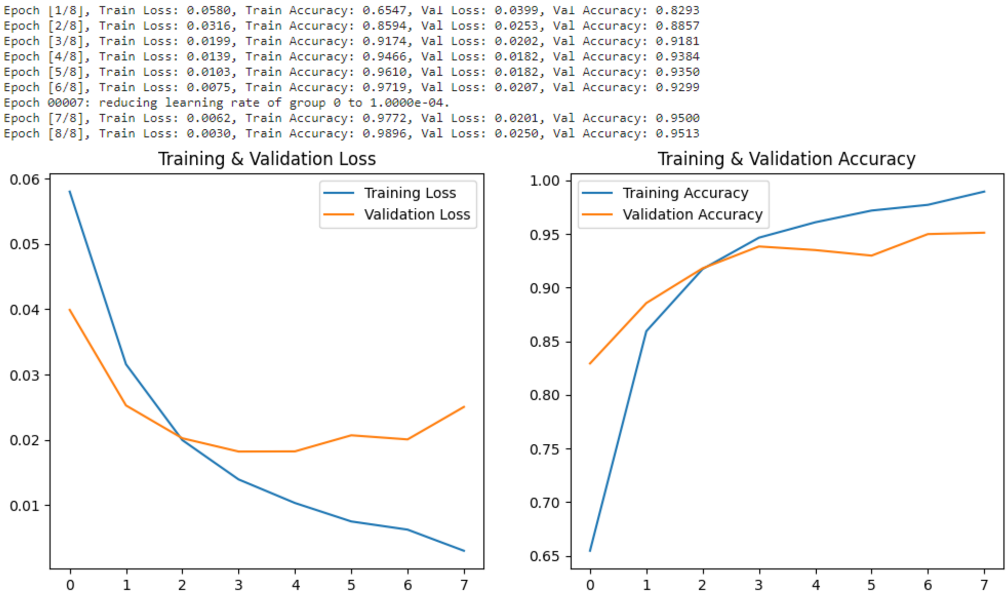


Figure 10: Training Log for CNNBiLSTM

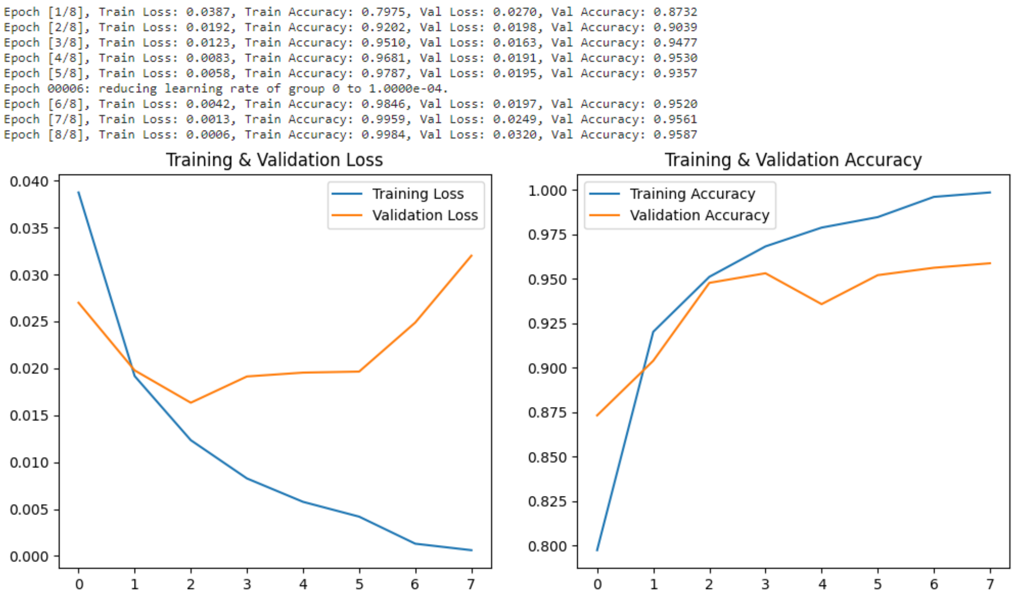


Figure 11: Training Log for Attention BiLSTM

```
Epoch [1/8], Train Loss: 0.0531, Train Accuracy: 0.6972, Val Loss: 0.0360, Val Accuracy: 0.7757
Epoch [2/8], Train Loss: 0.0365, Train Accuracy: 0.8253, Val Loss: 0.0257, Val Accuracy: 0.8912
Epoch [3/8], Train Loss: 0.0283, Train Accuracy: 0.8715, Val Loss: 0.0211, Val Accuracy: 0.8891
Epoch [4/8], Train Loss: 0.0230, Train Accuracy: 0.9010, Val Loss: 0.0182, Val Accuracy: 0.9174
Epoch [5/8], Train Loss: 0.0185, Train Accuracy: 0.9228, Val Loss: 0.0188, Val Accuracy: 0.9009
Epoch [6/8], Train Loss: 0.0149, Train Accuracy: 0.9407, Val Loss: 0.0165, Val Accuracy: 0.9442
Epoch [7/8], Train Loss: 0.0120, Train Accuracy: 0.9533, Val Loss: 0.0173, Val Accuracy: 0.9433
Epoch [8/8], Train Loss: 0.0099, Train Accuracy: 0.9632, Val Loss: 0.0189, Val Accuracy: 0.9400
```

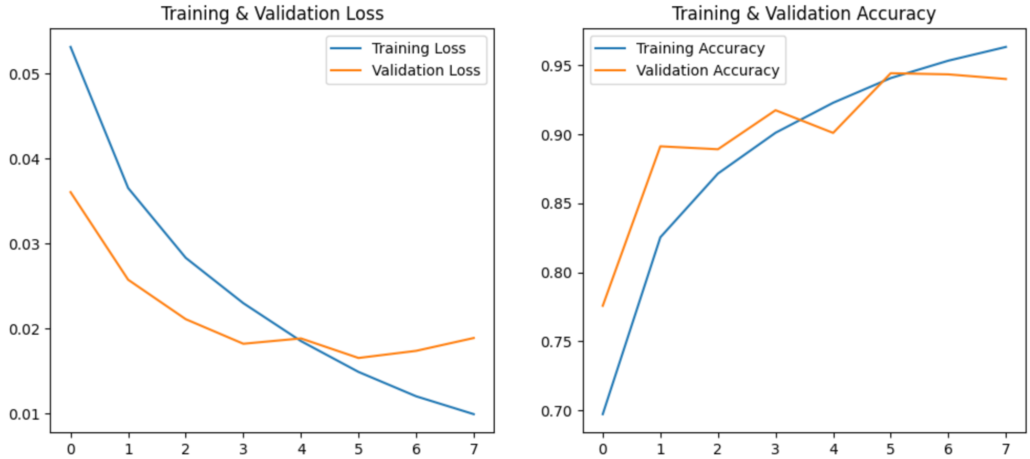


Figure 12: Training Log for CNNBiLSTMAttention

```
Epoch [1/8], Train Loss: 0.0525, Train Accuracy: 0.7021, Val Loss: 0.0363, Val Accuracy: 0.7866
Epoch [2/8], Train Loss: 0.0358, Train Accuracy: 0.8279, Val Loss: 0.0270, Val Accuracy: 0.8558
Epoch [3/8], Train Loss: 0.0278, Train Accuracy: 0.8745, Val Loss: 0.0228, Val Accuracy: 0.8716
Epoch [4/8], Train Loss: 0.0229, Train Accuracy: 0.9021, Val Loss: 0.0182, Val Accuracy: 0.9285
Epoch [5/8], Train Loss: 0.0187, Train Accuracy: 0.9214, Val Loss: 0.0187, Val Accuracy: 0.9129
Epoch [6/8], Train Loss: 0.0149, Train Accuracy: 0.9397, Val Loss: 0.0168, Val Accuracy: 0.9253
Epoch [7/8], Train Loss: 0.0120, Train Accuracy: 0.9546, Val Loss: 0.0190, Val Accuracy: 0.9188
Epoch [8/8], Train Loss: 0.0097, Train Accuracy: 0.9629, Val Loss: 0.0158, Val Accuracy: 0.9467
```

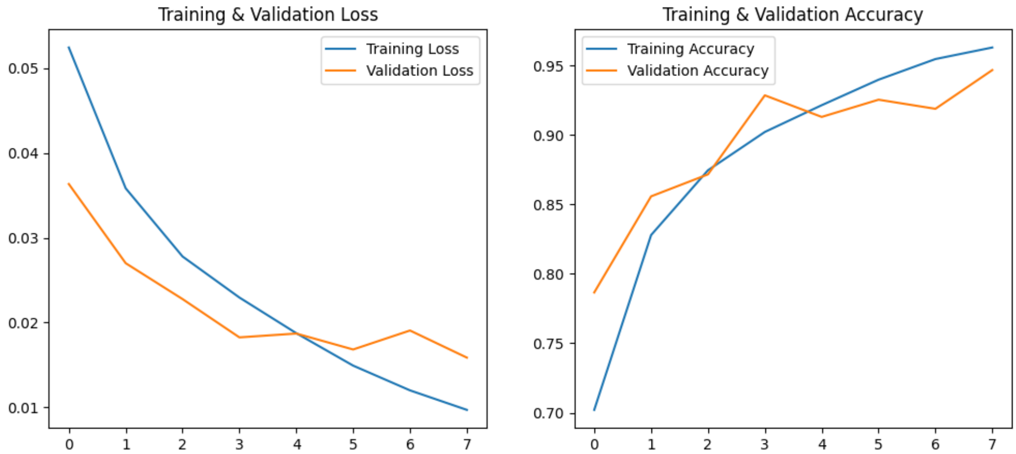


Figure 13: Training Log for CNNBiLSTMDouAttention

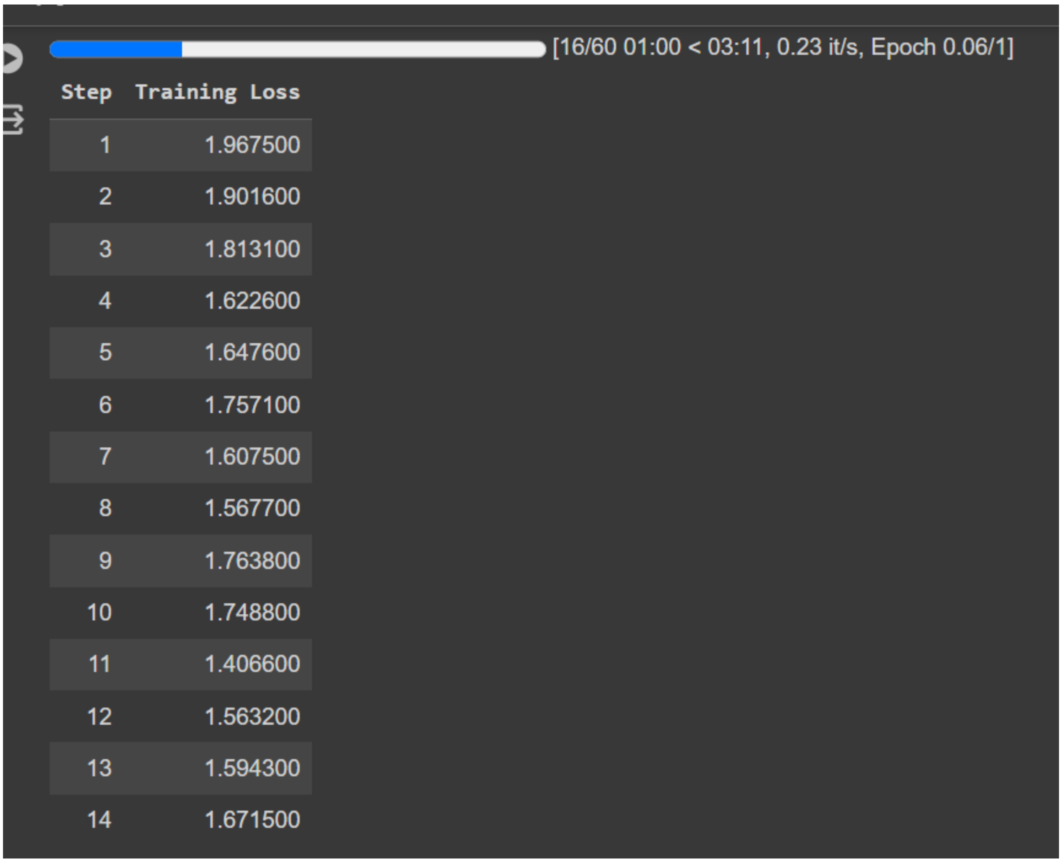
5.3.3 Mistral-7B

[!htbp] In the SFTTrainer, it already show the training loss inside each epoch.

5.4 Evaluation Metrics

In the testing process, I will record the model different metrics like the accuracy, F1 score, recall and the auc for knowing the overall performance of the model.

Meanwhile, a domain-specific metric and the generator-specific metric will be recorded to understand the model's power in different source of the domain. Moreover, the data of its performance of



[16/60 01:00 < 03:11, 0.23 it/s, Epoch 0.06/1]

Step	Training Loss
1	1.967500
2	1.901600
3	1.813100
4	1.622600
5	1.647600
6	1.757100
7	1.607500
8	1.567700
9	1.763800
10	1.748800
11	1.406600
12	1.563200
13	1.594300
14	1.671500

Figure 14: Training Log for Mistral-7B

identifying the text generating from different model will also be recorded to know the performance over it, especially on the human-text.

In the context of the MixSet dataset, rather than extracting domain-specific or generator-specific metrics, we applied mixing-method-specific metrics to evaluate the models' performance across different data mixing techniques based on the criteria from 5. This approach allows for a direct assessment of each model's ability to handle variably mixed data, essential for applications in diverse real-world scenarios.

5.5 Results

5.6 Caption of Different Models in the figures

Figure 15: Caption for the result section

5.6.1 SVM

Table 3: SVM Domain Specific performance In M4 Dataset

Model	wikihow	arxiv	wikipedia	peerread	reddit
Linear	0.87	0.77	0.89	0.92	0.86
Poly	0.90	0.82	0.91	**0.97**	0.89
RBF	**0.91**	**0.84**	**0.93**	0.96	**0.91**
Sigmoid	0.83	0.73	0.85	0.84	0.82

Table 4: SVM Generator Specific performance in M4 Dataset

Model	human	cohere	chatGPT	davinci
Linear	0.91	0.84	0.95	0.86
Poly	0.94	0.91	**1.00**	0.90
RBF	**0.96**	0.90	0.99	**0.91**
Sigmoid	0.85	0.79	0.91	0.82

Table 5: Performance Matrix in M4 dataset

Model	Accuracy	Precision	Recall	F1	AUC
Linear	0.86	0.86	0.86	0.86	0.93
Poly	0.90	0.90	0.90	0.90	**0.97**
RBF	**0.91**	**0.91**	**0.91**	**0.91**	**0.97**
Sigmoid	0.81	0.81	0.81	0.81	0.89

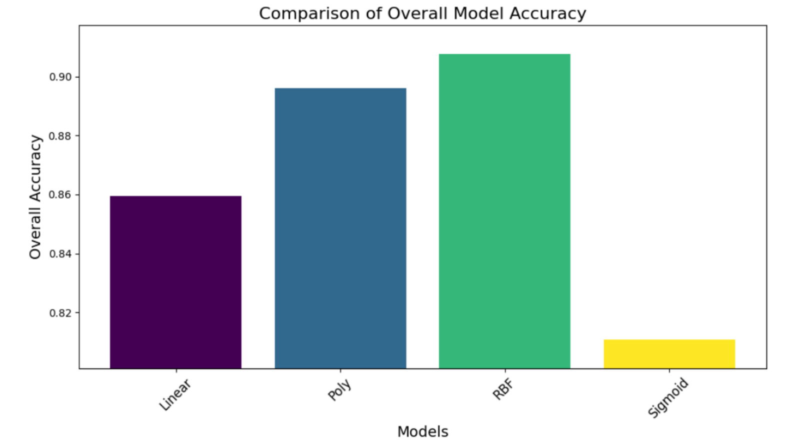


Figure 16: The SVM accuracy of different models in M4 dataset

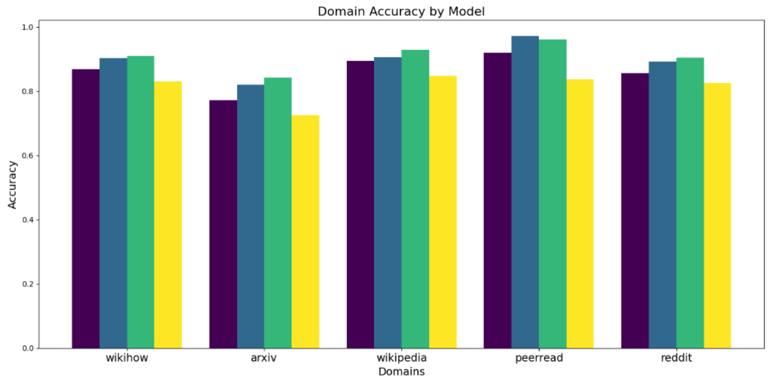


Figure 17: The SVM domain specific accuracy in M4 dataset

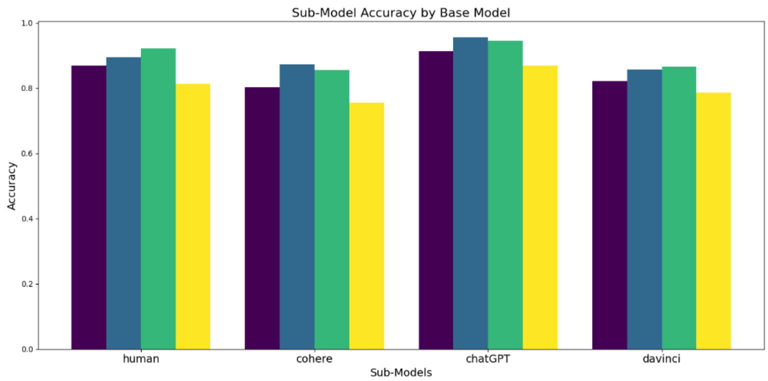


Figure 18: The SVM generator specific accuracy in M4 dataset

Table 6: SVM domain specific performance in MGTBench dataset

Model	story	essay	news
Linear	0.77	**0.69**	0.84
Poly	0.78	0.63	**0.89**
RBF	**0.79**	0.68	**0.89**
Sigmoid	0.75	0.67	0.82

Table 7: SVM generator specific performance in MGTBench dataset

Model	human	GPT4	chatGPT
Linear	0.79	0.94	0.90
Poly	0.72	**1.00**	**0.97**
RBF	**0.79**	0.97	0.94
Sigmoid	0.77	0.90	0.87

Table 8: SVM performance matrix in MGTBench dataset

Model	Accuracy	Precision	Recall	F1	AUC
Linear	0.77	0.77	0.77	0.77	0.85
Poly	0.77	0.78	0.77	0.76	0.83
RBF	**0.79**	**0.79**	**0.79**	**0.78**	**0.87**
Sigmoid	0.75	0.75	0.75	0.75	0.83

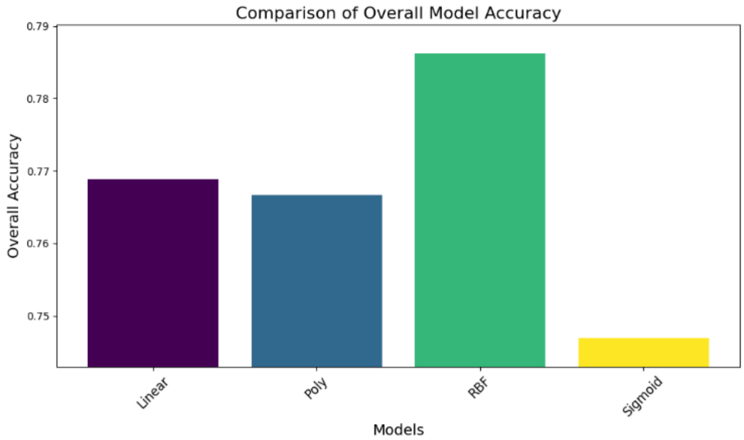


Figure 19: The SVM accuracy of different models in MGTBench dataset

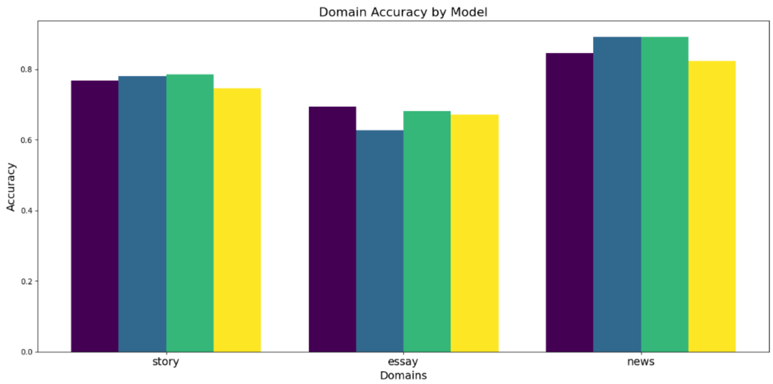


Figure 20: The SVM domain specific accuracy in MGTBench dataset

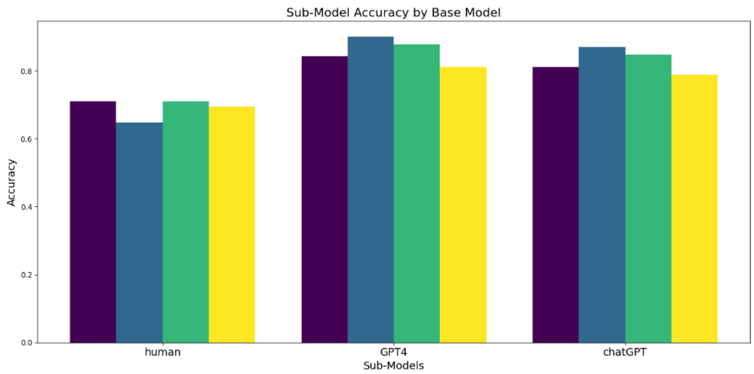


Figure 21: The SVM generator specific accuracy in MGTBench dataset

Table 9: The SVM mixing method accruacy in MixSet

Model	Humanize	Complete	Rewrite	Polish
Linear	**0.64**	**0.02**	0.04	0.06
Poly	0.56	0.02	0.04	0.06
RBF	**0.64**	**0.02**	0.01	0.02
Sigmoid	0.59	0.03	**0.06**	**0.09**

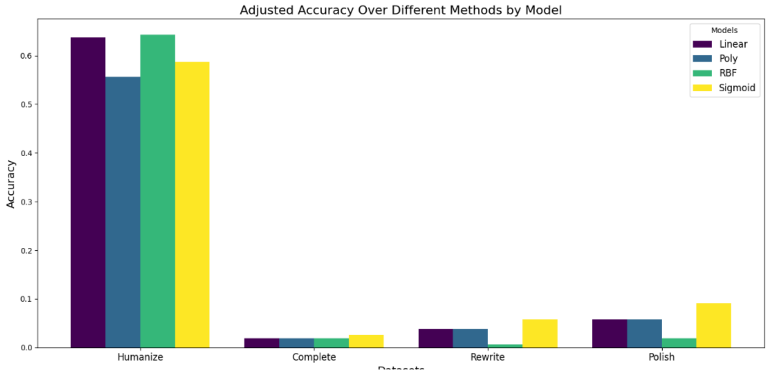


Figure 22: The SVM Mixing method accuracy in MixSet dataset

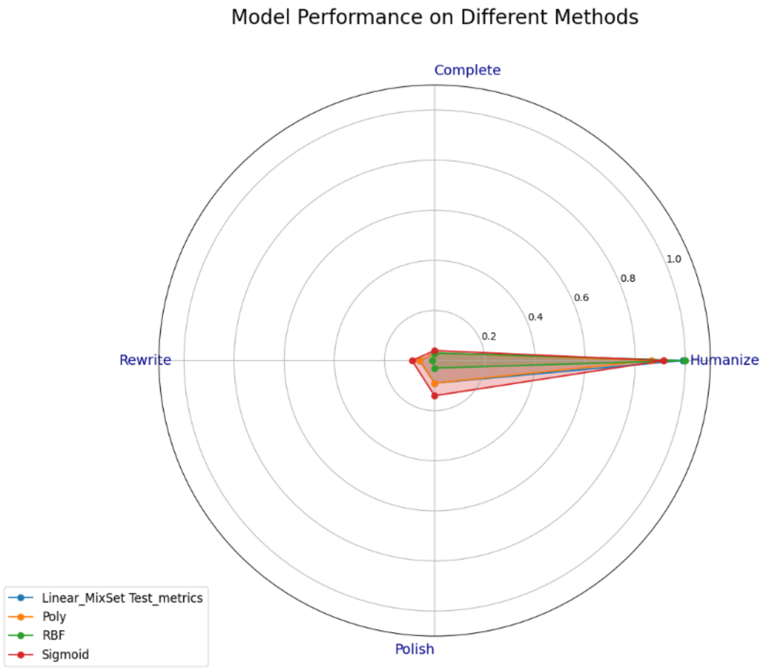


Figure 23: The SVM radar chart in MixSet dataset

5.7 Caption of Different Models in the figures

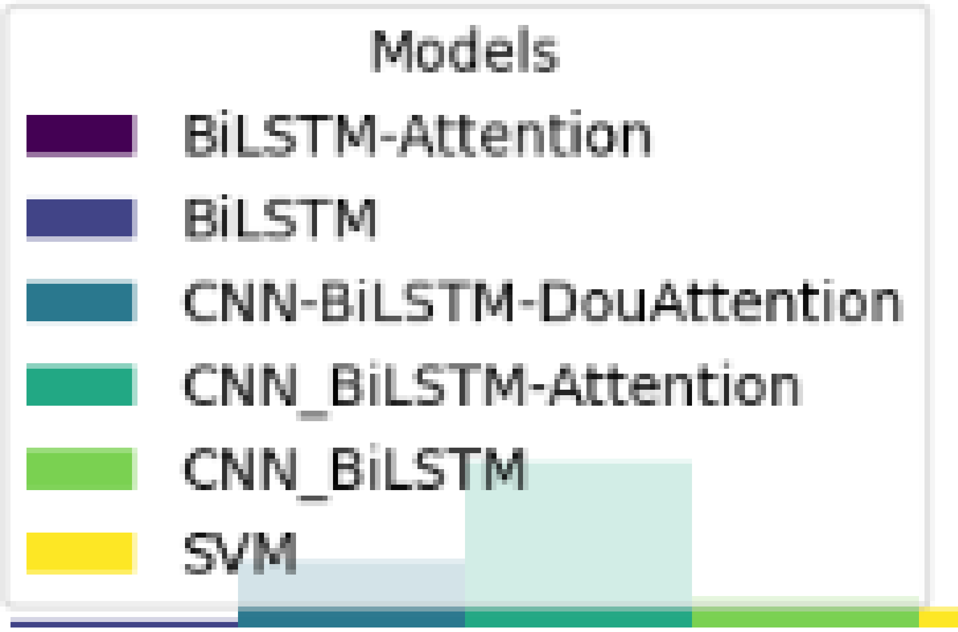


Figure 24: Caption for the result section

5.7.1 BiLSTM series

For BiLSTM series result, they will also compared with SVM with Poly kernel as the baseline.

Table 10: Domain Specific performance In M4 Dataset

Model	WikiHow	ArXiv	Wikipedia	PeerRead	Reddit
BiLSTMAttention	**0.96**	**0.95**	**0.94**	**0.99**	**0.96**
BiLSTM	0.95	0.94	0.93	0.99	0.95
CNNBiLSTMDouAttention	0.94	0.92	0.92	0.99	0.95
CNNBiLSTM	0.95	0.94	0.93	0.99	0.95
CNNBiLSTMAttention	0.93	0.90	0.89	0.98	0.93
SVM	0.83	0.73	0.85	0.84	0.82

Table 11: Generator specific performance in M4 dataset

Model	human	cohere	chatGPT	davinci
BiLSTMAttention	0.99	0.93	**0.99**	**0.92**
BiLSTM	0.99	0.91	0.98	0.90
CNNBiLstmDouAttention	0.99	0.90	0.98	0.87
CNNBiLSTM	0.99	0.91	**0.99**	0.89
CNNBiLSTMAttention	**1.00**	0.83	0.96	0.82
SVM	0.82	0.77	0.88	0.80

Table 12: Performance Matrix in M4 dataset

Model	Accuracy	Precision	Recall	F1	AUC
BiLSTMAttention	**0.96**	**0.96**	**0.96**	**0.96**	**0.96**
BiLSTM	0.95	0.95	0.95	0.95	0.95
CNNBiLstmDouAttention	0.94	0.95	0.94	0.94	0.94
CNNBiLSTM	0.95	0.95	0.95	0.95	0.95
CNNBiLSTMAttention	0.93	0.93	0.93	0.93	0.93
SVM	0.81	0.81	0.81	0.81	0.89

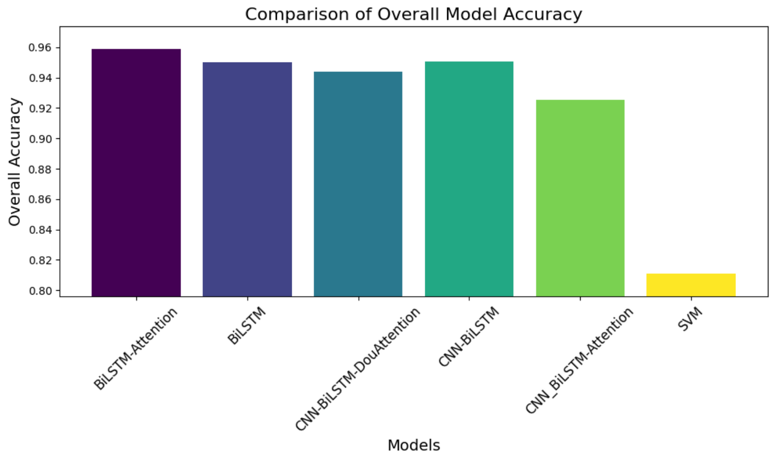


Figure 25: The accuracy of different models in M4 dataset

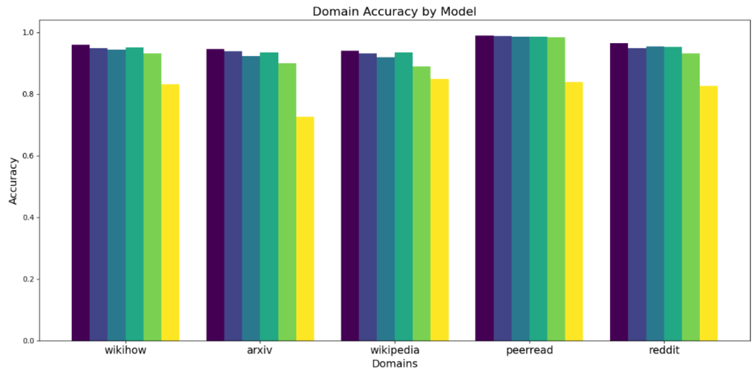


Figure 26: The domain specific accuracy in M4 dataset

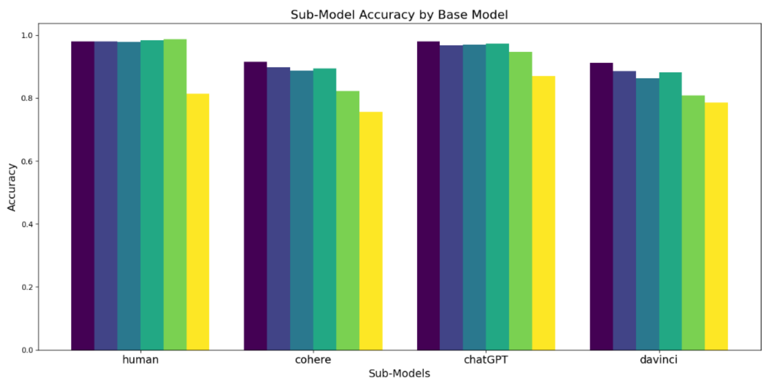


Figure 27: The generator specific accuracy in M4 dataset

Here are the results in MGTBench:

Table 13: Domain Specific performance in MGTBench

Model	story	essay	news
BiLSTMAttention	**0.81**	0.65	**0.87**
BiLSTM	0.77	0.67	**0.87**
CNNBiLstmDouAttention	0.78	0.69	0.83
CNNBiLSTMAttention	0.77	**0.73**	0.83
CNNBiLSTM	0.76	0.68	0.86
SVM	0.75	0.67	0.82

Table 14: Generator specific performance in MGTBench

Model	human	GPT4	chatGPT
BiLSTMAttention	0.77	**1.00**	**0.95**
BiLSTM	0.80	0.97	0.89
CNNBiLstmDouAttention	0.83	0.93	0.87
CNNBiLSTMAttention	**0.89**	0.89	0.82
CNNBiLSTM	0.82	0.94	0.87
SVM	0.78	0.91	0.89

Table 15: Performance Matrix in MGTBench

Model	Accuracy	Precision	Recall	F1	AUC
BiLSTMAttention	**0.78**	**0.79**	**0.78**	**0.78**	**0.78**
BiLSTM	0.77	0.77	0.77	0.77	0.77
CNNBiLstmDouAttention	0.77	0.77	0.77	0.77	0.77
CNNBiLSTMAttention	**0.78**	0.78	**0.78**	**0.78**	**0.78**
CNNBiLSTM	0.77	0.77	0.77	0.77	0.77
SVM	0.75	0.75	0.75	0.75	0.83

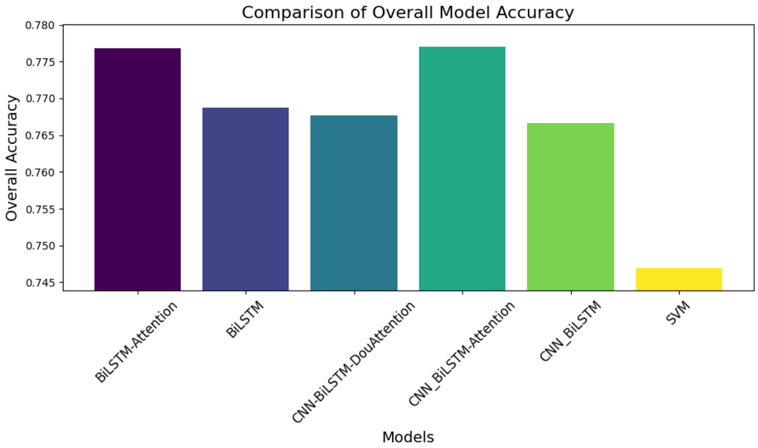


Figure 28: The accuracy of different models in MGTBench dataset

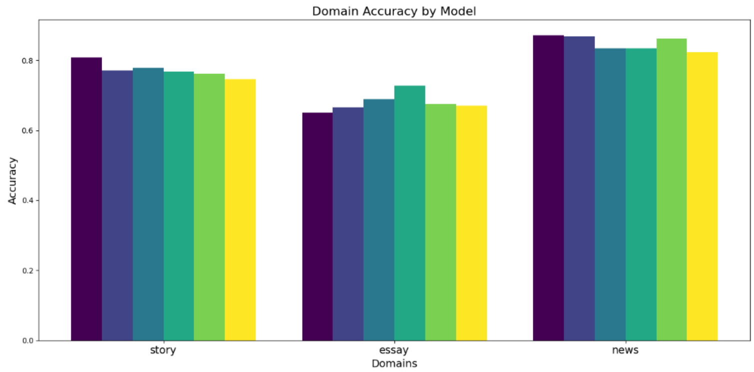


Figure 29: The domain specific accuracy in MGTBench dataset

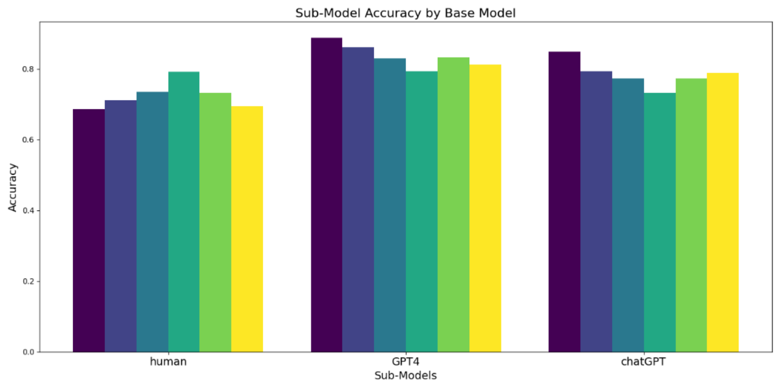


Figure 30: The generator specific accuracy in MGTBench dataset

Here is the result in MixSet:

Table 16: Mixing Method performance in Mixset

Model	Humanize	Complete	Rewrite	Polish
BiLSTMAttention	0.01	0.41	0.36	0.36
BiLSTM	0.01	**0.92**	**0.87**	0.91
CNNBiLSTMDouAttention	0.01	0.85	0.83	0.88
CNNBiLSTMAttention	0.01	0.90	0.86	**0.92**
CNN_BiLSTM	0.06	0.81	0.68	0.81
SVM	**0.56**	0.03	0.01	0.03

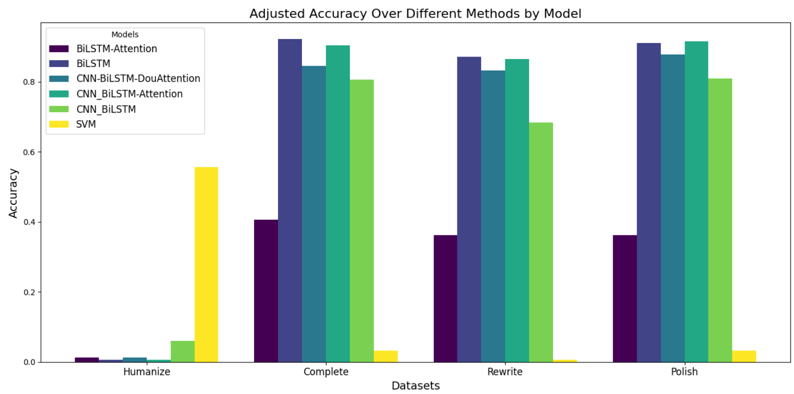


Figure 31: The Mixing method accuracy in MixSet dataset

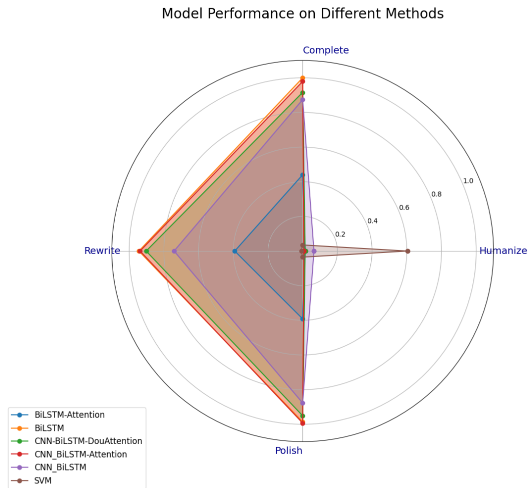


Figure 32: The radar chart in MixSet dataset

5.7.2 Mistral

Table 17: Mistral Overall Performance on the M4 Dataset

Metric	Value
Accuracy	0.6747
F1 Score	0.7429
Precision	0.6044
Recall	0.9636
AUC	0.6816
False Positives	5376

Table 18: Mistral Accuracy by Domain in M4 Dataset

Domain	Accuracy
Arxiv	0.5571
Peerread	0.9172
Reddit	0.7925
Wikihow	0.6123
Wikipedia	0.5218

Table 19: Mistral Accuracy by Model in M4 Dataset

Model	Accuracy
ChatGPT	0.9512
Cohere	0.9492
Davinci	0.9888
Human	0.3995

Table 20: Mistral Overall Performance on the MGTBench Dataset

Metric	Value
Accuracy	0.5473
F1 Score	0.6862
Precision	0.5243
Recall	0.9930
AUC	0.5486
False Positives	5374

Table 21: Mistral Accuracy by Domain in MGTBench Dataset

Domain	Accuracy
Essay	0.5322
News	0.5813
Story	0.5285

Table 22: Mistral Accuracy by Model in MGTBench Dataset

Model	Accuracy
GPT4	0.9885
ChatGPT	0.9973
Human	0.1043

Table 23: Mistral Accuracy by Dataset in MixSet

Dataset	Accuracy
Complete	0.00129
Humanize	1.0000
Polish	0.0581
Rewrite	0.0387

5.8 Analysis

5.8.1 SVM Kernel Compasison

In the evaluation presented in Section 5.6.1, it is clear that the RBF and Poly kernels substantially outperform the other kernels across various test datasets. Specifically, the RBF kernel achieves the highest scores in both Table 5 and Table 8. While the Poly kernel generally performs slightly worse than the RBF kernel, it shows superior performance in handling machine-generated data, notably outperforming the RBF kernel with Cohere and ChatGPT data in Table 4, and with ChatGPT and GPT-4 in Table 7. This indicates that the Poly kernel may be more effective in detecting nuances in machine-generated texts.

Why RBF Kernel Performs Better:

- **Non-Linear Decision Boundaries:** The RBF kernel excels because it can model complex, non-linear decision boundaries with greater flexibility. Unlike linear kernels, RBF kernels do not assume linear separability, which is particularly advantageous in natural language tasks where class boundaries are inherently non-linear and complex.
- **Handling High Dimensionality:** RBF kernels effectively deal with high-dimensional data (a common characteristic of text data), as they focus on the distance between support vectors rather than the dimensionality itself. This characteristic allows them to manage the curse of dimensionality better than linear models.
- **Versatility in Feature Mapping:** The RBF kernel maps samples into a higher-dimensional space more adeptly than polynomial or sigmoid kernels, making it superior at capturing relationships between class labels and features that are not readily apparent in the original space.

However, both kernels perform poorly with mixed data, as shown in Table 9 and Figure 23, achieving near-zero accuracy in three out of four mixing methods. This poor performance suggests that these kernels struggle to extract relevant features from mixed data, behaving almost randomly, particularly under the humanize mixing method where accuracy hovers around 60%.

5.8.2 BiLSTM series

M4 Dataset Analysis of the M4 dataset reveals that BiLSTM models generally surpass the SVM in performance due to their complex architecture and effective integration of LSTM, CNN, and Attention mechanisms, which are adept at capturing critical features in the content for classification (Table 12). Among the BiLSTM configurations, the BiLSTM-Attention model performs the best, underscoring the benefit of focusing on difficult-to-classify instances.

The domain-specific analysis in Table 10 shows variability in performance across different domains. Models perform nearly perfectly on PeerRead data but exhibit weaker performance on Wiki data, likely due to the diverse linguistic features and complex structures found in encyclopedic text.

Generator-specific analysis in Table 11 highlights a performance disparity between different types of generators. Models handle human or ChatGPT data effectively but struggle with Cohere and Davinci data, indicating a potential gap in training that fails to cover the linguistic patterns used by these generators.

MGTBench In MGTBench, the BiLSTM-Attention model again outshines other models, including SVM, as detailed in Table 15. However, there is a noticeable performance drop in this dataset compared to the M4 dataset, with accuracies falling from above 90% to around 80%. This drop likely reflects the challenges posed by out-of-distribution data, which differ significantly from the training data.

Domain analysis in Table 13 shows decent performance on narrative texts but poor results on essays, where the structured, argumentative nature of essays may not be well captured by the models. Surprisingly, the accuracy for detecting unseen GPT-4 generated data is exceptionally high, suggesting that these models are effectively generalizing from other data types to GPT-4's style.

However, the performance on human-written texts (Table 14) is lacking, contributing significantly to the overall drop in performance. This issue may stem from the models' inability to differentiate between human-like synthetic text and genuine human text, possibly due to varied linguistic styles and expressions that were not present in the training set.

MixSet For data that mixes machine-generated features, the BiLSTM series shows commendable performance in three out of four methods, except for the humanize method (Tables 16 and 32). Here, accuracy's reach up to 92%, indicating robustness in handling mixed data. Conversely, the SVM shows abysmal performance, further confirming its unsuitability for complex feature interactions found in mixed data.

The stark performance degradation in the humanize method for all models suggests that the GPT-4's deep reconstruction capabilities effectively camouflage the synthetic nature of texts, leading models to misclassify them as human-written.

5.8.3 Mistral-7B

The Mistral-7B model shows surprisingly low accuracy in both the M4 dataset and MGTBench (Tables 17 and 20), performing even worse than the SVM. However, its recall rates are competitive, indicating a tendency to correctly identify machine-generated texts but frequently mislabel human-written texts as machine-generated, a clear sign of overfitting to machine-text characteristics.

In the MixSet dataset (Table 23), Mistral-7B similarly performs poorly across most mixing methods except for the humanize method, where it achieves perfect scores. This anomaly may reflect an overbias toward machine-generated text characteristics, misinterpreting nuanced human-like features as indicative of machine origin.

6 Conclusion

This project has explored various methodologies for detecting machine-generated text across different datasets, domains, and generators. Notably, some methods demonstrated robust performance, even identifying out-of-distribution data such as that generated by GPT-4 with high accuracy. These successes underscore the capabilities of advanced models like BiLSTM-Attention and sophisticated kernels in SVMs to adapt and generalize from training data to unseen, novel text inputs.

Challenges Encountered: However, several challenges were encountered, particularly the tendency of models to misclassify human-written text as machine-generated. This issue is critical in applications such as academic plagiarism detection, where false positives can have significant negative consequences. The polynomial and RBF kernels in SVMs, while effective in many cases, showed limitations in their ability to discern complex patterns in mixed data, as evidenced by their poor performance on the MixSet dataset.

Implications of Findings: The use of GPT-4 to humanize original machine-generated data presented a unique challenge, as it was able to fool most models effectively. This finding highlights a potential vulnerability in current detection methods, which may struggle to cope with sophisticated techniques used in the latest generation of language models.

Future Directions: To overcome these challenges, future work should focus on:

- Enhancing the sensitivity of models to subtle linguistic cues that differentiate human and machine-generated texts.
- Developing more sophisticated data mixing techniques that can better simulate the nuanced characteristics of blended human-machine text.
- Exploring adaptive or dynamic modeling approaches that can adjust their strategies based on the nature of the text being analyzed.

Overall, this research has laid a solid foundation for further studies and highlighted critical areas for improvement in the detection of machine-generated text. The insights gained from the performance of various models on complex datasets such as M4 and MGTBench are invaluable for advancing the field of text analysis and enhancing the reliability of machine-generated text detection systems.

References

Unsloth AI. Unsloth: Open source code repository. https://github.com/unslothai/unsloth, 2024.

D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. https://arxiv.org/abs/1409.0473, 2014. arXiv preprint arXiv:1409.0473.

B. A. Becker, P. Denny, J. Finnie-Ansley, A. Luxton-Reilly, J. Prather, and E. A. Santos. Programming is hard–or at least it used to be: Educational opportunities and challenges of ai code generation. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, pages 500–506, 2023.

D. Chen. Mixset: Official code repository for mixset. https://github.com/Dongping-Chen/MixSet1, 2024. GitHub repository.

Y. Cheng, L. Yao, G. Zhang, T. Tang, G. Xiang, H. Chen, and Z. Cai. Text sentiment orientation analysis of multi-channels cnn and bigru based on attention mechanism. *Journal of Computer Research and Development*, 57(12):2583, 2020.

A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, et al. Palm: Scaling language modeling with pathways. https://arxiv.org/abs/2204.02311, 2022. ArXiv preprint abs/2204.02311.

J. Deng, L. Cheng, and Z. Wang. Attention-based bilstm fused cnn with gating mechanism model for chinese long text classification. *Computer Speech & Language*, 68:101182, 2021.

J. Devlin, M. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In J. Burstein, C. Doran, and T. Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 4171–4186, Minneapolis, MN, USA, 2019. NAACL-HLT. https://doi.org/10.18653/v1/n19-1423.

Y. Dong, P. Liu, Z. Zhu, Q. Wang, and Q. Zhang. A fusion model-based label embedding and self-interaction attention for text classification. *IEEE Access*, 8:30548–30559, 2019. doi: 10.1109/ACCESS.2019.2954985.

J. Du, L. Gui, R. Xu, and Y. He. A convolutional attention model for text classification. In *National CCF Conference on Natural Language Processing and Chinese Computing*, pages 183–195, Cham, 2017. Springer.

Hugging Face. 4-bit transformers with bitsandbytes. https://huggingface.co/blog/4bit-transformers-bitsandbytes, 2024a.

Hugging Face. Progressive effort fine-tuning (peft) technique. https://huggingface.co/blog/peft, 2024b.

Hugging Face. Sft trainer documentation. https://huggingface.co/docs/trl/sft_trainer, 2024c.

S. Giorgi, D. M. Markowitz, N. Soni, V. Varadarajan, S. Mangalik, and H. A. Schwartz. "i slept like a baby": Using human traits to characterize deceptive chatgpt and human text. In M. Litvak, I. Rabaev, R. Campos, A. M. Jorge, and A. Jatowt, editors, *Proceedings of the IACT - The 1st International Workshop on Implicit Author Characterization from Texts for Search and Retrieval*, volume 3477 of *CEUR Workshop Proceedings*, pages 23–37, Taipei, Taiwan. CEUR-WS.org. https://ceur-ws.org/Vol-3477/paper4.pdf.

B. Guo, X. Zhang, Z. Wang, M. Jiang, J. Nie, Y. Ding, J. Yue, and Y. Wu. How close is chatgpt to human experts? comparison corpus, evaluation, and detection. https://arxiv.org/abs/2301.07597, 2023. ArXiv preprint abs/2301.07597.

D. Gurkhe, N. Pal, and R. Bhatia. Effective sentiment analysis of social media datasets using naive bayesian classification. *International Journal of Computer Applications*, 975(8887):99, 2014.

X. He, X. Shen, Z. Chen, M. Backes, and Y. Zhang. Mgtbench: Benchmarking machine-generated text detection. https://arxiv.org/abs/2303.14822, 2023a. ArXiv preprint abs/2303.14822.

X. He, X. Shen, Z. Chen, M. Backes, and Y. Zhang. Mgtbench: Benchmarking machine-generated text detection. https://arxiv.org/abs/2303.148222, Mar 2023b. arXiv preprint arXiv:2303.14822.

A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, et al. Mistral 7b. https://arxiv.org/abs/2310.06825, 2023.

E. Kasneci, K. Seßler, S. Kuchemann, M. Bannert, D. Dementieva, F. Fischer, U. Gasser, G. Groh, S. Gunnemann, E. Hüllermeier, et al. Chatgpt for good? on opportunities and challenges of large language models for education. *Learning and Individual Differences*, 103:102274, 2023.

H. Lee, D. A. Hudson, K. Lee, and C. D. Manning. Slm: Learning a discourse language representation with sentence unshuffling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1551–1562. Association for Computational Linguistics, 2020. https://aclanthology.org/2020.emnlp-main.120.

J. Lee, T. Le, J. Chen, and D. Lee. Do language models plagiarize? In *Proceedings of the ACM Web Conference 2023*, pages 3637–3647, 2023.

P. Li, W. Xu, C. Ma, J. Sun, and Y. Yan. Ioa: Improving svm based sentiment classification through post processing. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 545–550, 2015.

G. Liang, J. Guerrero, and I. Alsmadi. Mutation-based adversarial attacks on neural text detectors. https://arxiv.org/abs/2302.05794, 2023. ArXiv preprint abs/2302.05794.

T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. https://arxiv.org/abs/1708.02002, Aug 2017. arXiv preprint arXiv:1708.02002.

G. Liu and J. Guo. Bidirectional lstm with attention mechanism and convolutional layer for text classification. *Neurocomputing*, 337:325–338, 2019.

Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. http://arxiv.org/abs/1907.11692, 2019.

Y. Ma, J. Liu, and F. Yi. Is this abstract generated by ai? a research for the gap between ai-generated scientific text and human-written scientific text. https://arxiv.org/abs/2301.10416, 2023. ArXiv preprint abs/2301.10416.

Y. Mirsky, A. Demontis, J. Kotak, R. Shankar, D. Gelei, L. Yang, X. Zhang, M. Pintor, W. Lee, Y. Elovici, et al. The threat of offensive ai to organizations. *Computers & Security*, page 103006, 2022.

A. Muñoz-Ortiz, C. Gómez-Rodríguez, and D. Vilares. Contrasting linguistic patterns in human and llm-generated text. https://arxiv.org/abs/2308.09067, 2023. ArXiv preprint abs/2308.09067.

M. S. Orenstrakh, O. Karnalim, C. A. Suarez, and M. Liut. Detecting llm-generated text in computing education: A comparative study for chatgpt cases. https://arxiv.org/abs/2307.07411, 2023. ArXiv preprint abs/2307.07411.

A. Pagnoni, M. Graciarena, and Y. Tsvetkov. Threat scenarios and best practices to detect neural fake news. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1233–1249. International Committee on Computational Linguistics, 2022. https://aclanthology.org/2022.coling-1.106.

B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. https://arxiv.org/abs/cs/0205070, 2002. arXiv preprint cs/0205070.

X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai, and X. Huang. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, 63(10):1872–1897, 2020.

V. S. Sadasivan, A. Kumar, S. Balasubramanian, W. Wang, and S. Feizi. Can ai-generated text be reliably detected? https://arxiv.org/abs/2303.11156, 2023. ArXiv preprint abs/2303.11156.

D. Shen, M. Zheng, Y. Shen, Y. Qu, and W. Chen. A simple but tough-to-beat data augmentation approach for natural language understanding and generation. https://arxiv.org/abs/2009.13818, 2020. ArXiv preprint abs/2009.13818.

Z. Shi and M. Huang. Robustness to modification with shared words in paraphrase identification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 164–171. Association for Computational Linguistics, 2020. https://aclanthology.org/2020.findings-emnlp.16.

Z. Shi, Y. Wang, F. Yin, X. Chen, K.-W. Chang, and C.-J. Hsieh. Red teaming language model detectors with language models. https://arxiv.org/abs/2305.19713, 2023. ArXiv preprint abs/2305.19713.

L. SiChen. A neural network based text classification with attention mechanism. In *2019 IEEE 7th*. IEEE, October 2019.

C. Stokel-Walker. Ai bot chatgpt writes smart essays should academics worry? *Nature*, 2022.

A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations*, New Orleans, LA, USA, 2019. ICLR.

G. Wang, C. Li, W. Wang, Y. Zhang, D. Shen, X. Zhang, and L. Carin. Joint embedding of words and labels for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2321–2331, 2018.

J. Wang, L. C. Yu, K. R. Lai, and X. Zhang. Dimensional sentiment analysis using a regional cnn-lstm model. In *Proceedings of the 54th annual meeting of the association for computational linguistics*, volume 2 of *Short papers*, pages 225–230, 2016.

Y. Wang, J. Mansurov, P. Ivanov, J. Su, A. Shelmanov, A. Tsvigun, C. Whitehouse, O. M. Afzal, T. Mahmoud, A. F. Aji, and P. Nakov. M4: Multi-generator, multi-domain, and multi-lingual black-box machine-generated text detection. https://arxiv.org/abs/2305.149024, May 2023. arXiv preprint arXiv:2305.14902.

Y. Wu and G. Deng. Interactive attention network fusion bi-lstm and cnn for text classification. In *Proc. SPIE 12254, International Conference on Electronic Information Technology (EIT 2022)*, page 122542F. International Society for Optics and Photonics, 2022. https://doi.org/10.1117/12.2638585.

Y. Xiao, Y. Li, J. Yuan, S. Guo, Y. Xiao, and Z. Li. History-based attention in seq2seq model for multi-label text classification. *Knowledge-Based Systems*, 224:107094, 2021.

Z. Yang, Z. Dai, Y. Yang, J. G. Carbonell, R. Salakhutdinov, and Q. V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alche Buc, E. B. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 5754–5764, Vancouver, BC, Canada, 2019. NeurIPS. https://proceedings.neurips.cc/paper/2019/hash/dc6a7e655d7e5840e66733e9ee67cc69-Abstract.html.

A. Yuan, A. Coenen, E. Reif, and D. Ippolito. Wordcraft: Story writing with large language models. In *27th International Conference on Intelligent User Interfaces*, pages 841–852, 2022.

Q. Zheng, X. Xia, X. Zou, Y. Dong, S. Wang, Y. Xue, Z. Wang, L. Shen, A. Wang, Y. Li, et al. Codegeex: A pre-trained model for code generation with multilingual evaluations on humanevalx. https://arxiv.org/abs/2303.17568, 2023. ArXiv preprint abs/2303.17568.