Introduction

With the rise of generative models like ChatGPT and GPT-4, there are increasing trend of the usage of AI in different field like academic writing, generating stories, code generation (cite 7 - 9) because the writing ability of LLMs have reached a comparable level to human writing (Survey of AI-text Generation cite 1, 2, 3).

The power of generation capabilities of LLMs will comes with its own set of challenges. There has a difficulty for individuals to identify the AI-generated text and human-written texts, leading to different concerns. Those concerns are mainly from three parts. The first one is the quality of the information generated, because of the limitation of the training data feed during the training process, LLM can only rely on those outdated information and heightened sensitivity to prompts, which will generate the wrong information, having plagiarism if not review carefully because of that. Another point is the misusing of the power of LLMs in different areas like generating fake news, spam information and even in essay writing as well. (Cite 24, 25, 23, 19, 18)

Various research has been proposed on the detection of machine-generated text, especially with the rise of ChatGPT. The current-state-of-the-art approach are using fine-tuning different language models with a large training data for the classification task. However, that research are mainly assuming one fact: all data used are either purely human-written or machine-generated, which is deprived from the real world situation as people may told LLM to refine their written version of word or they may told LLM to make the machine-generated result more closely to what human would write.

This project will tackle the challenge of differentiating between AI-generated and human-written texts. The approach to then problem will be a exploration of existing methodologies with the proposal of the innovative detection techniques which is combining LSTM with CNN and Attention. The project, like the other also hypothesis, a good understanding of textual characteristics, combined with advanced deep learning methods, can significantly improve the ability to identify between the AI or human text. The project will also focus on their performance on text length, model generating the AI-text and the mixed AI-human content to know their power in different situations.

The significance of this research lies not only in its immediate application to academic and creative writing but also in its broader implications for the future of AI

in society. As we delve deeper into the realm of artificial intelligence, establishing reliable methods for identifying AI-generated content becomes crucial for ethical standards, copyright laws, and the overall transparency of information. Through this project, we expect to contribute valuable insights and tools to the ongoing discourse on AI's role in content creation and its implications for human creativity and authenticity.

Related Work

1. Text Classification

Text classification is one of the most fundamental and popular work in natural language processing, and has been a research focus. The task is to understand the semantic meaning of the given sentences or paragraphs and label them based on the need. It has been applied in different areas such as fake news detection and machine-text detection. Meanwhile, there were traditional machine learning methods mainly doing the feature extraction using bag-of-words or TF-IDF such as using N Naïve Bayes, Decision Trees, and Support Vector Machines (SVMs) [1-3] (Cite LSTM-CNN-Interative-Attention 1-3).

However, with the rapid development of LLMs, detecting the machine-text will be a more challenging problem due to the fact that LLM-generated text is very closed to human-written text in terms of human-habit and their writing style. In fact, some studies showed that human performance in identifying LLM-generated text and human-text are closer to random classification only (Need prove).

Even though LLM-generated text is more align to human-writing, some studies have highlighted some significant point for identifying the text as LLM-generated in terms of lengthy text, excess reasoning and vocabs or in terms of formality which make the result are more like an expert giving answer rather than an ordinary person when answering the questions [Survey 40-43]. In view of this, our project will mainly be focusing on doing machine-text classification. Various methods would be used to detect rather it is LLM-generated text or human text.

2. LSTM model

The field on text classification had been shifted because of the introduction on deep neural networks like CNNs and even RNNs [CNN-RNN-LSTM-ATTENTION model]. Unlike those traditional methods focusing on feature extraction, CNNs and RNNs had build an big improvement as they can capture the important patterns in text by

themselves directly without feature engineering. For example, CNNs can capture the local text features and RNNs can address the sequence dependencies.

Nonetheless, people have found another path for combining the advantage in using CNNs and RNNs in order to make the more comprehensive features inside the text sequences which can improve both accuracy [CNN-RNN-ATTENTION 17-23]. Notably, developments such as embedding label sets into vector spaces for computation by G. Wang et al. [29], and the incorporation of self-attention and label-embedding techniques by Dong, Y et al. [30] and Y. Xiao et al. [31] For example, an study combining CNN and Attention as the encoder with the BiLSTM as the decoder for doing the stock prediction, while another paper combining CNN and BiLSTM with an interactive Attention for extracting the features and semantics of the text at the same time in fake news detection. The hybrid models could enhanced the interpretabiliy and make it more capable in text classification, especially if the LLM-generated text are more similar to human. In the project, rather combining the CNN-BiLSTM model with interactive attention, it will more closer to the () model adding the attention between CNN and BiLSTM for doing an better feature extraction.

3. Fine-tuning LLM

With the advance of transformer acritecture and ChatGPT, there are many research focusing on utilizing the power of LLMs as the detector in text classification. Some work (125 in survey) showed that pre-training LLMs will be powerful in natural language understanding. In fact, current trend of pre-trained models are BERT [126], Roberta [127] and XLNet [128] as they showed superior performance in different benchmark. For example, Roberta showed the best detectors on LLM-generated text. And OpenAI [87] detector also adopting a Roberta fine-tuning approach. Nonetheless, most of the models shown above are with at least 7B parameters because of capturing more features in a large set of data, but they are difficult to let user with less GPU resources to fine-tune for their own task. In the project, it will try to use a much smaller LLM which is Mistral-7B with Lora to do the fine-tuning process.

4. Potential Attacks

Nonetheless, several research showed that even the model can identify the machine-text with high performance, with some different techniques can affect their performance. For example, using a lightweight paraphrase model on machine-text outputs to change the distribution of wording and meaning a bit could be effective on some zero-shot detection (150,151). Shi et al. [112] and He et al. [69] reported on

the effectiveness of the permutation approach like doing cutoff (153), shuffle (154), mutation (155) and word swapping (156) on attack detectors. In our work, we will using the dataset of MixSet (), where using the LLM to generate the human-machine mixture model by doing approach like humanizing the machine output as the paraphrase attacks on the models tested.

Data

Approach

1. Data Preparation

Before loading the datasets for training and testing process, prepare the data from the dataset first. It first need to remove all the dataset where its language is not English in the dataset. Then, it need 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 need to be separated into the training, validation and test dataset in 8:1:1 ratio for loading the datasets.

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.

3. Model Architecture

In the following part, I will explain the following models used: SVM, CNN-BiLSTM with attention and Mistral-7B

3.1 SVM

3.1.1 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 TF-IDF to converts the text data into a matrix of TF-IDF features to enabling the SVM to know the textual features. The TF-IDF 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.

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.

3.2 CNN-BiLSTM with attention

3.2.1 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.

3.2.2 Convolutional Layers

After converting to the word vectors, it will be applied into multiple convolutional layers. Each of them consists of a set of filters or kernels that slide across the word vectors to detect specific features or patterns at different position of the text. Assume the input text 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 convolutional layer that employs kernels of varying size in 2, 3, and 4 words to capture different local textual features. Those kernels will focus on different n-gram combinations while the global ax pooling reduces the feature map into condensed representation.

3.2.3 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.

3.2.4 Bi-directional LSTM

Receiving the attended feature vector from the Pre-LSTM Attention. 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.

3.2.5 Post-LSTM Attention Mechanism

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 emphasizes 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).

3.2.6 Comparing Baselines

For the comparison experiment, the following models with similar complexity were used:

- 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
- CNN-BiLSTM-Attention: A similar approach to the proposed model but removing the attention layer between CNN and BiLSTM to test the effectiveness of that layer.

3.3 Minstra-7B-bnb-4bit

3.3.1 General Model Architecture

Another model going to be used is Mistral 7B, according to their paper released, this model completely outperforms Llama 2 13B, the popular model in LLM field on all benchmarks, and even outperforms Llama 34B on many benchmarks, showing the small LLM’s ability is comparable with large ones with proper settings. In the project,

the Mistral-7B's variant which is block-wise model-update Filtering and Bit-centering (BNB) is used for enhancing the model efficiency and memory usage. Moreover, the quantized 4bits version would be used for reducing the model's size and can be trainable even in T4 GPU.

3.3.2 Training Setting

In the project, it will use 'FastLanguageModel' from UnSLoth library for downloading the model and setting the maximum sequence for up to 2048 tokens. Meanwhile, LoRA would be used to train only 4% of its parameters with gradient accumulation and precision training.

After the data loaded, rather applying standard natural language processing techniques like what I did in SVM and LSTM model, the Supervised Fine-Tuning method would be performed where the text data and their labels will be structured as a suitable prompt format for the model retraining on the Machine-text classification task.

3.3.3 Comparing Baseline

A zero-shot version of another LLM model Roberta would be compared by directing inputting the test data into the model to classify the task.

4. Training Loss

Preliminary Results

Data distribution

Since loading the complete M4 dataset is too time-consuming and too troublesome to do the fixing if there are any bugs inside any code, I have just loaded around 45k data in M4 dataset, splitting some of them as training, validation and testing data to do the classification. Here are the distribution of the data:

Model training setting

1. **SVM**

Since I can just directly calling the SVM models by using libraries, there are no additional settings other than the TF-IDF mentioned in above section.

2. **CNN-BiLSTM-DoubleAttention**

For the model and its variants, I have used the same initialization parameters to ensure they have similar complexity, here is the list of them:

They are trained with 10 epochs with AdamW with learning rate of 0.001, weight decay of 0.005 and the scheduler of ReduceLROnPlateau to train the models.

3. Mistral-7b-bnb-4bit

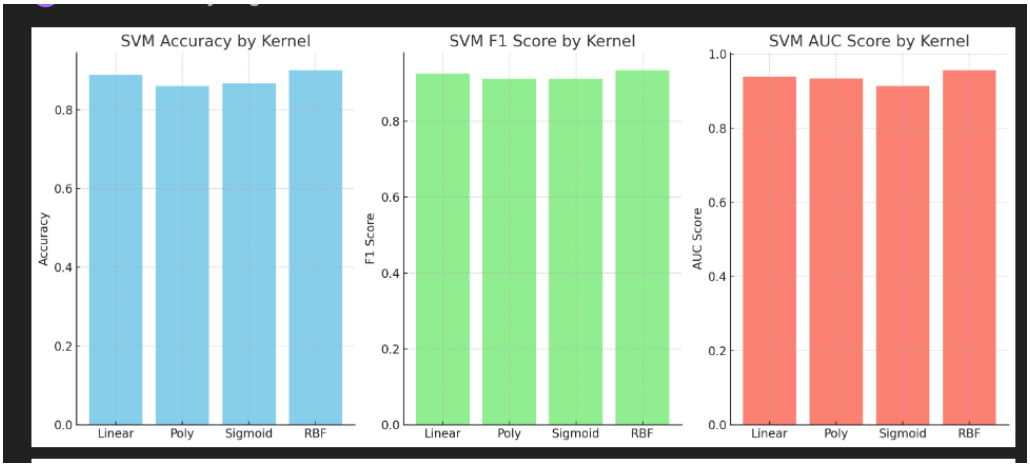
The most of the training settings are mentioned in the above sections, and this model is trained with Adamw 8bit versions with learning rate of 2e-4 and weight decay of 0.01. There are also linear learning rate schedular.

Results

SVM

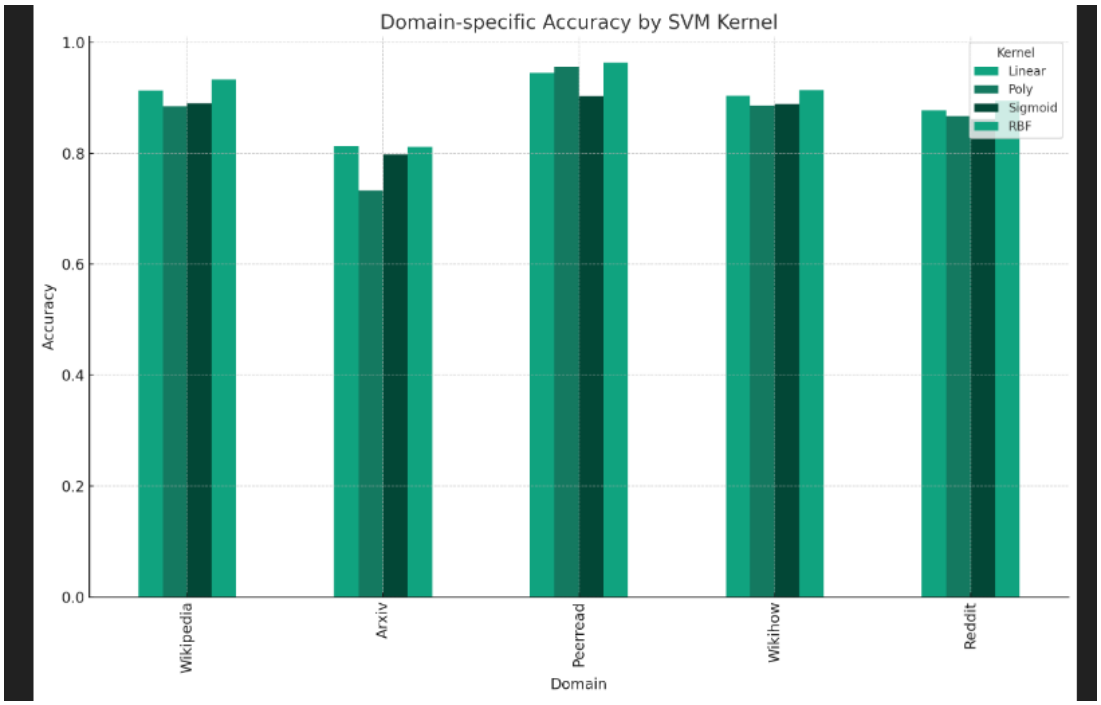
Metrics

	Linear	Poly	Sigmoid	RBF
Accuracy	0.887797781	0.860520095	0.866612111	**0.900436443**
F1 Score	0.925894787	0.911889719	0.911674393	**0.934654174**
AUC Score	0.938373151	0.933698033	0.914235356	**0.956517463**



Domain

	Linear	Poly	Sigmoid	RBF
Wikipedia	0. 913004882	0. 884598313	0. 890368398	**0. 933422104**
Arxiv	**0. 812836439**	0. 732505176	0. 797929607	0. 811594203
Peerread	0. 945089757	0. 956177402	0. 902851109	**0. 963041183**
Wikihow	0. 903604359	0. 886001676	0. 888935457	**0. 914082146**
Reddit	0. 877073171	0. 866829268	0. 861951220	**0. 895121951**



From those tables and results figures, it is clearly shows that SVM models with RBF kernel outperforms the others in terms of accuracy, F1 score and AUC score, showing it power in handling the dataset.

In terms of domain, RBF kernel shows consistent and nearly all the best performance over each domain. While some kernel like Poly kernel even performed well on Peeread domain, varied a lot on other domain. Showing those kernels are not comparable to RBF kernel in terms of consistency.

CNN-BiLSTM with double attention

Here are the results of this model and its variations

For simplicity

BiLSTM -> Model A

CNNBiLSTM -> Model B

AttentionBiLSTM -> Model C

CNNBiLSTM with Attention -> Model D

CNNBiLSTM with Double Attention -> Model E

General Metric

	Model A	Model B	Model C	Model D	Model E
accuracy	0.86147	0. 871704	**0. 89339**	0.88325	0.852923
F1 score	0.864837	0.875738	**0.893401**	0.88632	0.85916
precision	0.8718700	0.886632	**0.89340**	0.89396446	0.879765

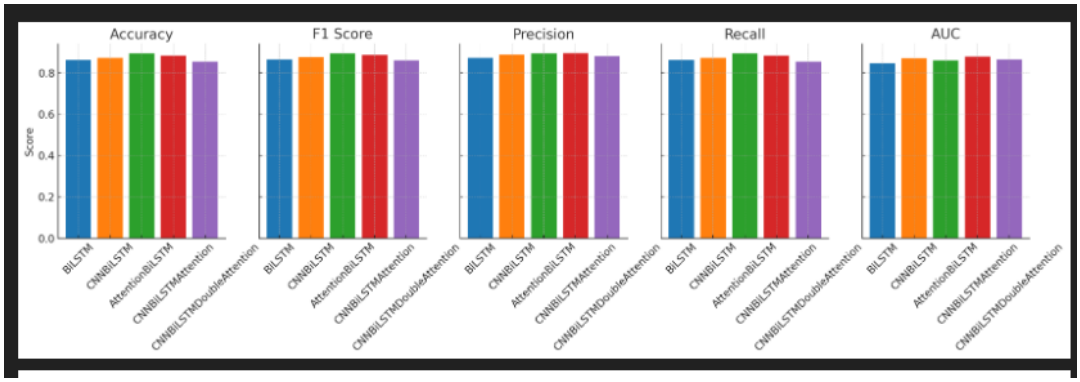
recall	0.861476	0.87170	**0.89339**	0.8832554	0.85292
auc	0.84647	0.87024	0.8602775	**0.8777912**	0.86460

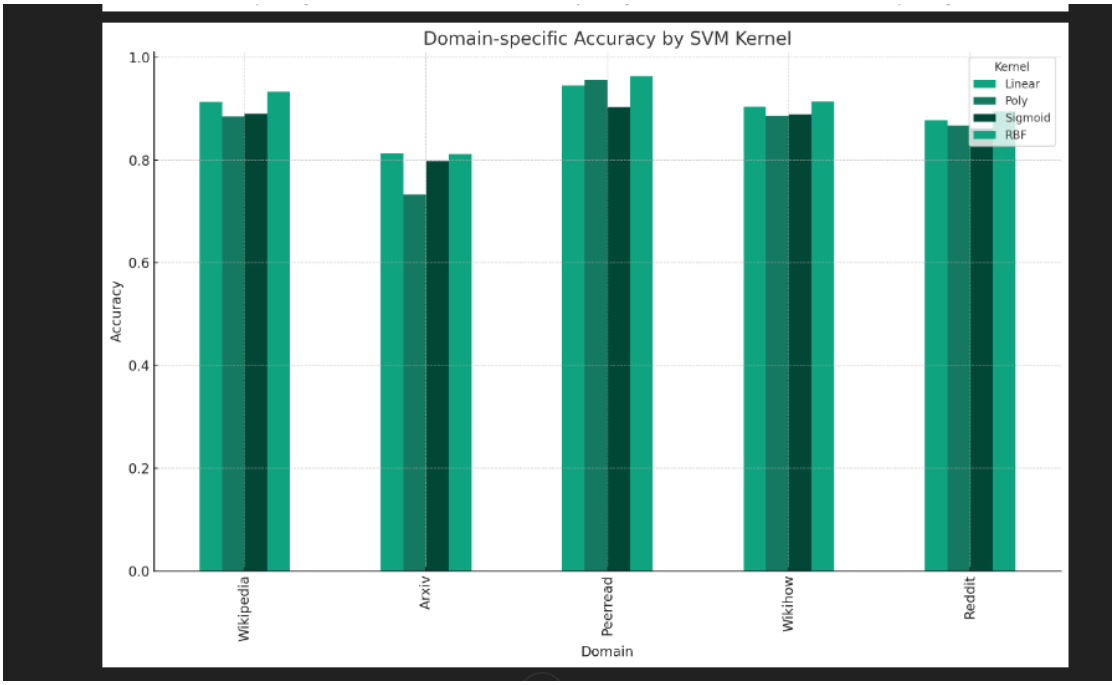
Accuracy in terms of Domain

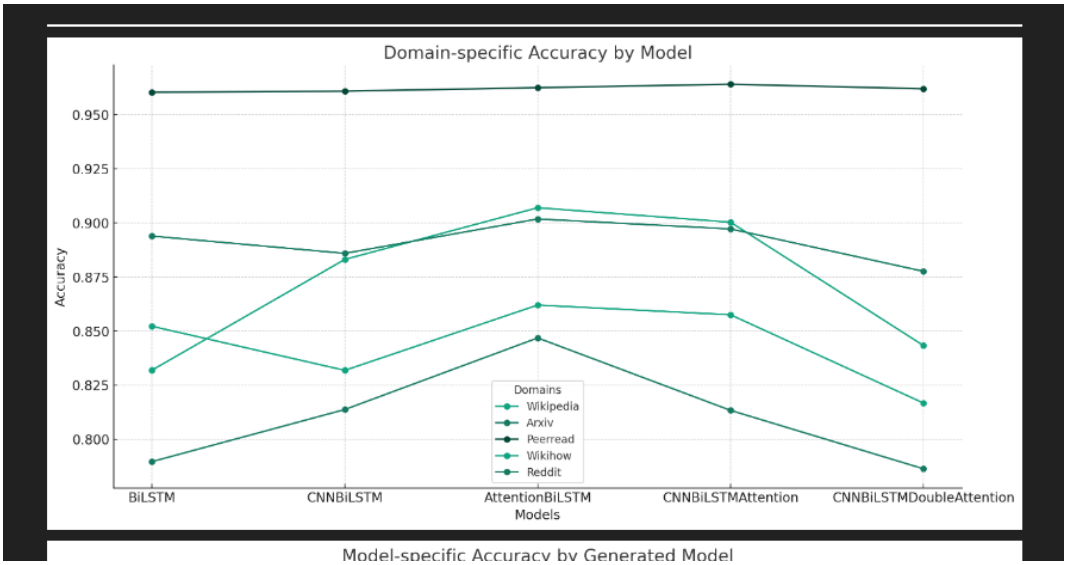
	Model A	Model B	Model C	Model D	Model E
Wikipedia	0.852197	0.831779	**0.86196182**	0.8575233	0.8166888
Arxiv	0.7896480	0.813664	**0.84679089**	0.8132505	0.786335
Peerread	0.96040126	0.96092925	0.96251319	**0.96409714**	0.9619852
wikihow	0.83193629	0.88306789	**0.9069572**	0.90025146	0.843252
reddit	0.8938570	0.88591725	**0.9017969**	0.89720016	0.8775595

Accuracy in terms of source of generating the text

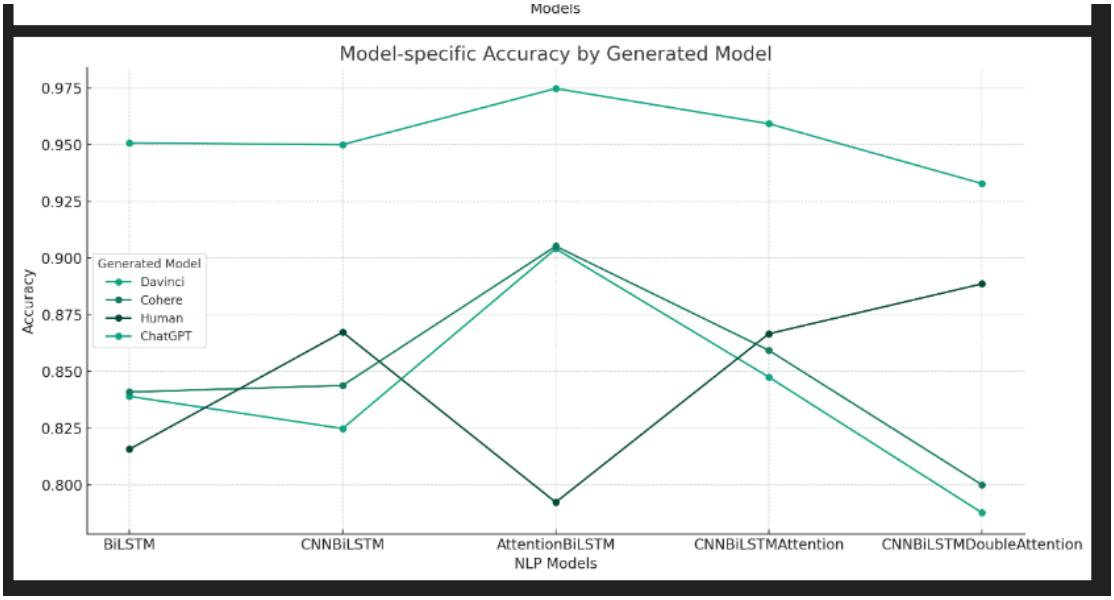
	Model A	Model B	Model C	Model D	Model E
davinci	0.83899504	0.82484076	**0.90410474**	0.84748761	0.7876857
cohere	0.840867	0.843761301	**0.9052441**	0.859312	0.8
human	0.81568088	0.86726272	0.7922971	0.8665749	**0.8885832**
chatGPT	0.95073891	0.95003518	**0.97466572**	0.9591836	0.932793







Model-specific Accuracy by Generated Model



In terms of the overall model performance, the AttentionBiLSTM (model C) showcases superior performance in terms of basically all the metrics against other models.

In terms of domain-specific accuracy, still the AttentionBiLSTM model consistently shows high accuracy across most domains, especially in Peerread and Wikihow domains. While others vary levels of effectiveness across different sources.

In terms of source generating the text, even though AttentionBiLSTM performed well when detecting different LLM model text, it performed poorly when detecting human text. However, the CNNBiLSTM with double attention perform consistent in terms of that. In detecting the text generated by ChatGPT, all the models performed well in it, showcasing that those models can greatly capture the features of whether it is generated by ChatGPT.

Reason for such results:

1. The data are greatly imbalanced, the number of human text and machine-text are in 1:3, meaning that those models can perform great results just by capturing the AI generated text well. (Even using the focal loss to deal with this problem)
2. Overfitting, in the experience, I have trained all the models in 10 epochs, in fact, some models like CNNBiLSTM with double attention, already overfit when running epoch 7 to 8, losing its performance all running that much epochs. It will be somehow relieved if using the complete size of dataset.

Mistral-7b-bnb-4bit

Even though I have completed the code to fine-tune the model, successfully without any bug, I don't have time trying to train the model with that dataset because I will otherwise late submit the results (Maybe appending the results after the submission as the appendix).

Roberta

I have not implemented the code to do the zero-shot of this model yet. Though it is easy to do so because I will just be doing the inferencing to test the zero-shot power of it.

Obstacles and next steps

For the obstacles, there has a huge problem when training the models with an imbalanced dataset as mentioned before. Moreover, even limiting the vector sized, the time needed to train all the SVM with different kernel are too time-consuming, it took 4.5 hours to train 4 kernels with just the part of the dataset in M4, worrying that it will take longer if larger data is used.

For the next step, I will think about how to tidy the data first to become more balanced for the training process. Meanwhile, I will try inventing other dataset for doing the out-of-distribution test in terms of the source or the model generating the text. And testing the model's performance in AI-Human mixture text. For the model's itself, I will try reducing the size of those model to preventing the overfitting issue as well.

- page report, clarifying the following things
- Title, Author(s)
- (20%) Introduction. Introduce your problem, and the landscape for why the problem is interesting and what has been done before in this space. Describe your overall plan for approaching the problem, what contributions you expect to make, and why this is interesting in the context of the described landscape.
- (25%) Related Work. Describe in detail existing work related to your problem, how they are related to each other, and how your work relates to these. We expect this to be comprehensive and thorough, and with at least 10 citations discussed and cited accordingly.
- (15%) Data. Describe in detail the data that you are using, including the source(s) of the data, relevant statistics, and qualitative examples if appropriate.
- (20%) Approach. Thoroughly describe the methods that you intended to use in your

approach, and the baselines you plan to compare against.
- (20%) Preliminary Results. Describe any preliminary results up to the time of the milestone. You should show the results of training at least one deep learning model. (It is fine if the model has not yet attained good performance.) You should also show preliminary analysis on the model(s) that you have trained. You should also describe anticipated next steps and any obstacles that have come up