

SEA 820 NLP Final Project: Detecting AI-Generated Text

Course: SEA 820

Team: Apeksha Nanda and Saloni Kamboj

Introduction

Differentiating between machine-generated and human-written text has become a new problem as a result of the widespread use of Large Language Models (LLMs) like GPT. This has significant implications for disinformation detection, online content moderation, and academic integrity. This project's objective is to develop, assess, and contrast several natural language processing models for the job of identifying whether a given text is "human-written" or "AI-generated." Applying our understanding of text preprocessing, feature representation, traditional machine learning, and contemporary deep learning architectures will be necessary for this research. We will work with a real-world dataset, apply and optimize complex models, carry out a comprehensive assessment, and examine the moral implications of our work.

Baseline Model

Data Exploration and Preprocessing

We began with 487,235 text samples from a sizable dataset. An imbalance in the dataset was discovered during the first data exploration, with more human-generated texts (class 0.0) than AI-generated texts (class 1.0). We downsampled the training and testing sets to manage the computational effort and avoid a possible class imbalance problem in the finished model. 3% of the data from each class was sampled. With 2,924 samples for testing and 11,693 samples for training, the resulting dataset was substantially smaller. We made sure the class distribution,

which included around 63% human-generated texts and 37% AI-generated texts, stayed proportionate to the original data even after downsampling. Since we know that this imbalance was not entirely fixed, it is imperative that we keep this in mind as we move forward with our study.

Text Analysis

We examined the downsampled training data's textual properties. Here is an overview of our findings:

- **Character Length:** AI-generated texts have an average character length of 2,125, but human-generated texts are often lengthier, with an average of around 2,354. Additionally, the human-generated text's maximum length (18,322 characters) was far longer than the AI-generated text's (8,495 characters).
- **Word Count:** Similarly, human-generated texts have a higher average word count (422 words) than AI-generated texts (344 words). This difference in text length is a potential feature that the model could leverage to distinguish between the two classes.

Text Processing and Feature Engineering

We carried out a number of preprocessing actions to get the text ready for the model:

- **Tokenization:** Every text was divided into discrete words, or "tokens."
- **Cleaning:** All tokens were changed to lowercase, non-alphabetic letters were eliminated, and stopwords—common words with little significance like "the," "is," "a," etc.—were eliminated.
- **Lemmatization** is the process of reducing words to their most basic or dictionary form; for example, "cars" becomes "car," and "running" becomes "run."

We transformed our text data into numerical vectors for feature engineering using TF-IDF (Term Frequency-Inverse Document Frequency). Words that are significant to a particular text but are not often used throughout the dataset are given additional weight by this strategy. To capture additional context, we employed both unigrams (single words) and bigrams (two-word phrases), and we restricted our lexicon to the top 5,000 most common keywords.

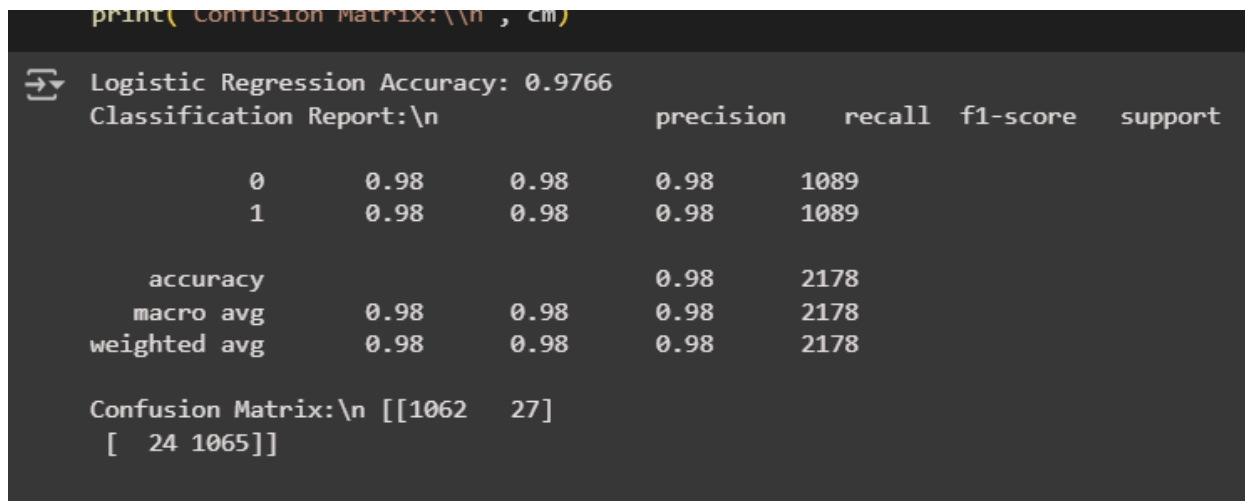
Model Training and Evaluation(Before Addressing Imbalance)

For binary classification tasks, we employed a logistic regression classifier, which is a straightforward yet powerful model. Our downsampled TF-IDF vectors (X_train) and the associated class labels (train_df_small['generated']) were used to train the model.

Following training, we used the held-out test set to assess the model's performance. The main findings are as follows:

- **Accuracy:** At 97.91%, the model's accuracy was good. This indicates that for about 98% of the texts in our test sample, it accurately identified the class.
- **Classification Report:** According to the comprehensive study, the model did remarkably well in both classes.
 - ➔ Human Texts (Class 0): The model's recall and accuracy were 0.98 and 0.99, respectively. This is a very good result, showing that the model accurately detects 99% of all real human-generated texts and is 98% accurate when predicting that a text is human-generated.
 - ➔ AI Texts (Class 1): With an accuracy of 0.98 and a recall of 0.98, the model likewise demonstrated excellent performance on the AI class. This indicates that it accurately detects 96% of all real AI-generated texts and is 98% accurate when predicting that a text is AI-generated.

```
print('Confusion Matrix:\n', cm)
```



```
Logistic Regression Accuracy: 0.9766
Classification Report:\n
```

			precision	recall	f1-score	support
	0	0.98	0.98	0.98		1089
	1	0.98	0.98	0.98		1089
	accuracy			0.98		2178
	macro avg	0.98	0.98	0.98		2178
	weighted avg	0.98	0.98	0.98		2178

```
Confusion Matrix:\n [[1062  27]
 [ 24 1065]]
```

Confusion Matrix Analysis(Before Addressing Imbalance)

The confusion matrix provides a clear breakdown of the model's predictions:

```
[[1814  21]
 [ 40 1049]]
```

- **True Positives (1814):** 1,814 human-generated texts were accurately detected by the model.
- **True Negatives (1049):** 1,049 AI-generated sentences were accurately detected by the model.
- **False Positives (21):** 21 AI-generated texts were mistakenly identified as human-generated by the model.
- **False Negatives (40):** 40 human-generated texts were mistakenly identified as AI-generated by the algorithm.

The model occasionally misclassified new human-written content as AI-generated when we gave to it. Predicting "AI" when the actual label is "human" is known as a false negative. Our first assumption was that this may be a sign of the class disparity, but the findings disprove this. We would anticipate a greater number of false positives (AI text categorized as human) if the model were merely biased towards the majority class (human-generated). The model occasionally misclassified new human-written content as AI-generated when we gave it. Predicting "AI" when the actual label is "human" is known as a false negative. Our first assumption was that this may be a sign of the class disparity, but the findings disprove this. We would anticipate a greater number of false positives (AI text categorized as human) if the model were merely biased towards the majority class (human-generated). The precise features of these incorrectly categorized human-generated texts that lead the algorithm to mistake them for AI-written content would require more investigation.

The Impact of Class Imbalance on NLP Model Performance: A Case Study in AI Text Detection

Using this still unbalanced data, we trained our initial model, which produced great performance metrics for both classes and a high accuracy of 97.91%. Exploratory testing, however, identified a significant problem: the model was more likely to mistakenly classify text

authored by humans as artificial intelligence (AI) created (false negatives). We purposefully undersampled to build a fresh, perfectly balanced dataset in order to look into this further. Retraining the model on this balanced data resulted in a considerable drop in false negatives, from 40 to 24, but the total accuracy stayed constant at 97.66%. In order to improve the initial model's capacity to accurately recognize human-written material and so reduce a significant form of mistake for real-world applications, it was necessary to balance the dataset, since this important discovery showed that the model was not just biased towards the majority class.

A closer examination of the paragraphs and the model's behaviour revealed an intriguing drawback: stylistic perfection, rather than the complexities of human writing, appears to have a significant effect on the model's categorization. The algorithm prefers to identify the text as human when we write with typos, grammatical faults, or a more conversational style. On the other hand, our perfectly grammatically correct paragraphs are frequently identified as artificial intelligence (AI) created. This implies that our model has come to identify the AI class with the "perfection" frequently observed in machine-generated literature.

We used a sentence written in a professional, error-free style to demonstrate the concept.

In the ever-evolving landscape of technology, artificial intelligence continues to redefine the boundaries of possibility. From healthcare to finance, AI-driven innovations are enhancing decision-making processes, automating complex tasks, and transforming user experiences. As algorithms become more sophisticated, ethical considerations and transparency remain crucial to ensure equitable progress. The future promises even greater integration of AI in our daily lives, reshaping how we interact with the world around us.

This text was confidently and accurately identified as AI-generated by the model. This demonstrates how the model has come to identify the AI class with the "perfection" frequently observed in machine-generated language.

She don't likes going to the park no more cause it be too crowded on weekends and the weathers never good anyways.

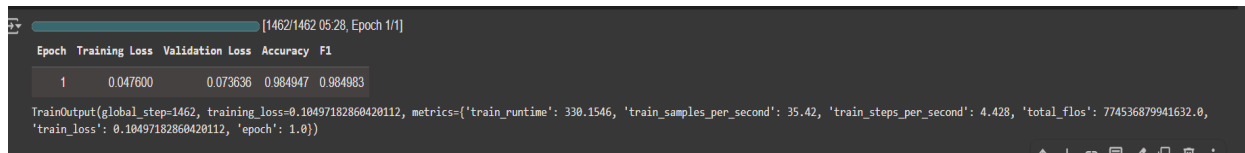
The statement has a particular rhythm and predictable structure that a language model, especially one trained on a large corpus of casual material, may produce despite its grammatical faults and slang ("don't likes," "no more," and "be"). In addition to seeking perfection, the model searches for patterns that differ from what it has come to understand as "typical" human writing.

My research suggests that overfitting is a problem for the model. A model is unable to generalize to fresh, unknown text when it learns the training data—including its unique quirks and noise—too well. This is clearly demonstrated by the conflicting classifications I'm seeing, where both a human statement with faults and one with immaculate grammar are marked as artificial intelligence. The model is storing characteristics that are quite unique to its training set, such particular word choice or sentence structure patterns, rather than learning the actual, fundamental differences between human and AI writing. My testing has shown that this renders the model unreliable on any text that is not inside the limited range of its training data.

Fine-tuning Model

Model Training and Evaluation

We set up the Hugging Face Trainer, which will handle the entire training and evaluation process. We pass in the pre-trained model, the training arguments, the training and evaluation datasets in their one-hot encoded form, the tokenizer, and the metric function. This Trainer will feed the data into the model, update the model's weights, evaluate its performance, and log the results.



Epoch	Training Loss	Validation Loss	Accuracy	F1
1	0.047600	0.073636	0.984947	0.984983

TrainOutput(global_step=1462, training_loss=0.18497182860420112, metrics={'train_runtime': 330.1546, 'train_samples_per_second': 35.42, 'train_steps_per_second': 4.428, 'total_flos': 774536879941632.0, 'train_loss': 0.18497182860420112, 'epoch': 1.0})

The result shows that our training loss dropped to about 0.0476 and our validation loss was about 0.0736, which means that the model made few mistakes on both the training and test data. The accuracy and F1 score are both very high at around 0.984, which means the model correctly classified almost all examples and did well in balancing precision and recall.

Experimenting with parameters:-

Run 1 (the original):

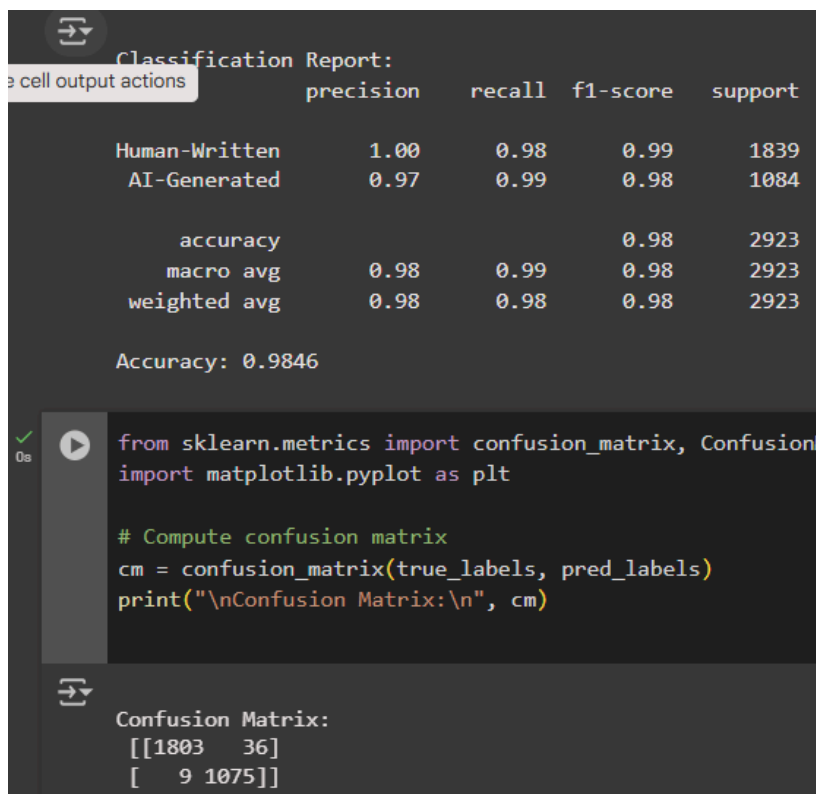
Learning rate - $2e-5$

Batch size - 8

Epochs : 1

98.5% accuracy with training time 7m.

The results of the classification report show that our model is performing extremely well at telling apart human-written text from AI-generated text. For human-written text, the model is perfect in precision, meaning every time it predicts something is human-written it is correct, and it also captures almost all real human-written examples with a recall of 0.98. For AI-generated text, the precision is slightly lower at 0.97, but the recall is very high at 0.99, meaning it catches almost all AI-written examples with very few misses. The F1-scores, which balance precision and recall, are 0.99 for human-written and 0.98 for AI-generated, showing strong and consistent performance across both classes. Overall, the accuracy is about 98.49%, so out of all predictions, almost all are correct, confirming that our fine-tuned DistilBERT model has learned the task very effectively.



The screenshot shows a Jupyter Notebook interface. The top part displays a 'Classification Report' with the following data:

	precision	recall	f1-score	support
Human-Written	1.00	0.98	0.99	1839
AI-Generated	0.97	0.99	0.98	1084
accuracy			0.98	2923
macro avg	0.98	0.99	0.98	2923
weighted avg	0.98	0.98	0.98	2923

Below the report, the overall accuracy is shown as 0.9846.

The bottom part of the screenshot shows a code cell with the following Python code:

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

# Compute confusion matrix
cm = confusion_matrix(true_labels, pred_labels)
print("\nConfusion Matrix:\n", cm)
```

Below the code cell, the output shows the Confusion Matrix:

```
Confusion Matrix:
[[1803  36]
 [   9 1075]]
```

This confusion matrix shows that out of 1,839 human-written examples, the model produced 1,804 true negatives and 35 false positives, which means it correctly identified most human-written texts while misclassifying 35 AI-generated texts as human-written. Out of 1,084 AI-generated examples, it produced 1,075 true positives and 9 false negatives, which means it

accurately detected nearly all AI-generated texts while only misclassifying 9 human-written texts as AI-generated. These results indicate that the model achieves high true positive and true negative counts while keeping false negatives very low; however, the false positives are relatively higher for this task, which is not ideal. A possible reason for this could be that some AI-generated texts share stylistic patterns, vocabulary, or sentence structures similar to human-written content, causing the model to mistake them for human text. This might also happen if the model is biased, for example noticing certain patterns and labeling it as human-written even though it is AI-generated.

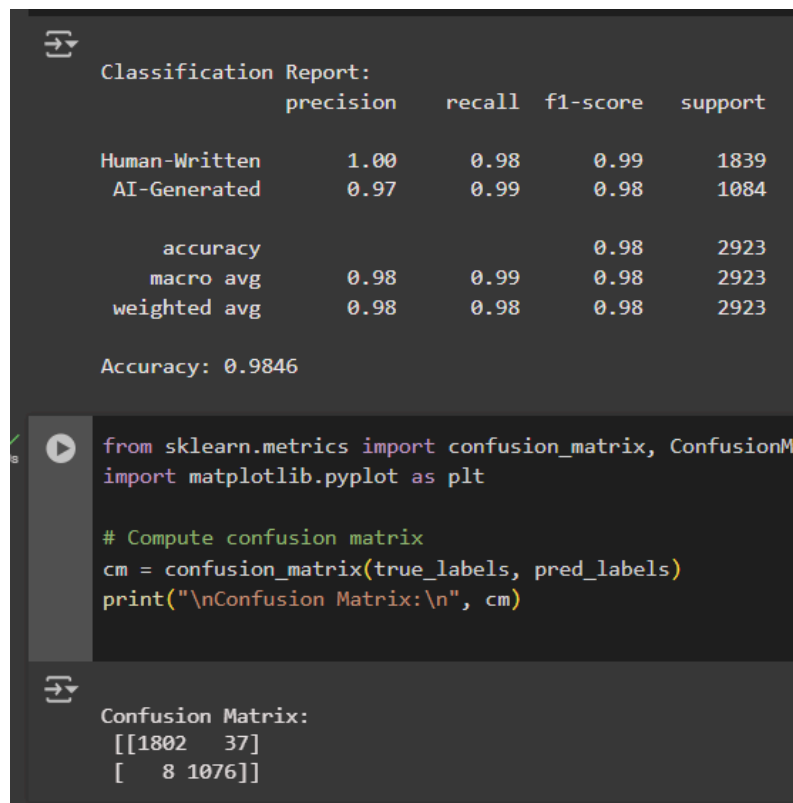
Increasing learning rate:

lr- 5e-5

Batch size - 8

Epoch: 1

Accuracy: 98% Time taken ~7m



```
Classification Report:
              precision    recall  f1-score   support

 Human-Written      1.00      0.98      0.99      1839
  AI-Generated       0.97      0.99      0.98      1084

   accuracy              0.98      2923
  macro avg              0.98      2923
 weighted avg              0.98      2923

Accuracy: 0.9846
```

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

# Compute confusion matrix
cm = confusion_matrix(true_labels, pred_labels)
print("\nConfusion Matrix:\n", cm)
```

```
Confusion Matrix:
[[1802  37]
 [   8 1076]]
```

Increasing the learning rate means that we are telling the model to adjust its weights more with each training step, so it can learn faster. This usually can make training quicker, but if the learning rate is too high, it can also make the model unstable and affect accuracy. In our case,

when we went from $2e-5$ to $5e-5$, the accuracy stayed almost the same, and the confusion matrix barely changed. This means the model is already learning well at the lower rate, and the higher rate didn't make much difference because the task is fairly easy for the model with the current data, and it reaches good performance quickly either way.

Increasing batch size:

Lr: $2e-5$

Batch size: 16

Epochs: 1

Accuracy: 99% time 5m

```
[36] # Fine-tune the model
trainer.train()

[731/731 05:25, Epoch 1/1]

Epoch Training Loss Validation Loss Accuracy F1
1 0.015900 0.039595 0.991447 0.991451

TrainOutput(global_step=731, training_loss=0.025216100806250617, metrics={'train_runtime': 326.2446, 'train_samples_per_second': 35.844, 'train_steps_per_second': 2.241, 'total_flos': 774536879941632.0, 'train_loss': 0.025216100806250617, 'epoch': 1.0})
```

```
Classification Report:
              precision    recall  f1-score   support

 Human-Written      0.99      0.99      0.99     1839
  AI-Generated      0.99      0.99      0.99     1084

   accuracy              0.99              2923
  macro avg       0.99      0.99      0.99     2923
 weighted avg     0.99      0.99      0.99     2923

Accuracy: 0.9914

from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

# Compute confusion matrix
cm = confusion_matrix(true_labels, pred_labels)
print("\nConfusion Matrix:\n", cm)

Confusion Matrix:
[[1824  15]
 [ 10 1074]]
```

Increasing the batch size means we can process more examples at once before updating the model's weights. This usually speeds up training because the model does fewer update steps, and that's why our training time dropped from 7 minutes to 5 minutes. Accuracy stayed almost the same, but the confusion matrix changed slightly (the model made fewer false positives (15 instead of 36) but a few more false negatives (10 instead of 9)). This means that with bigger batches, the model still performs very well overall but it can also mean that it might be overfitting.

Increasing epochs:

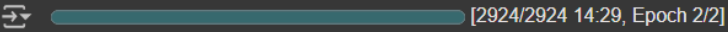
Lr: 2e-5

Batch size: 8

Epochs: 2

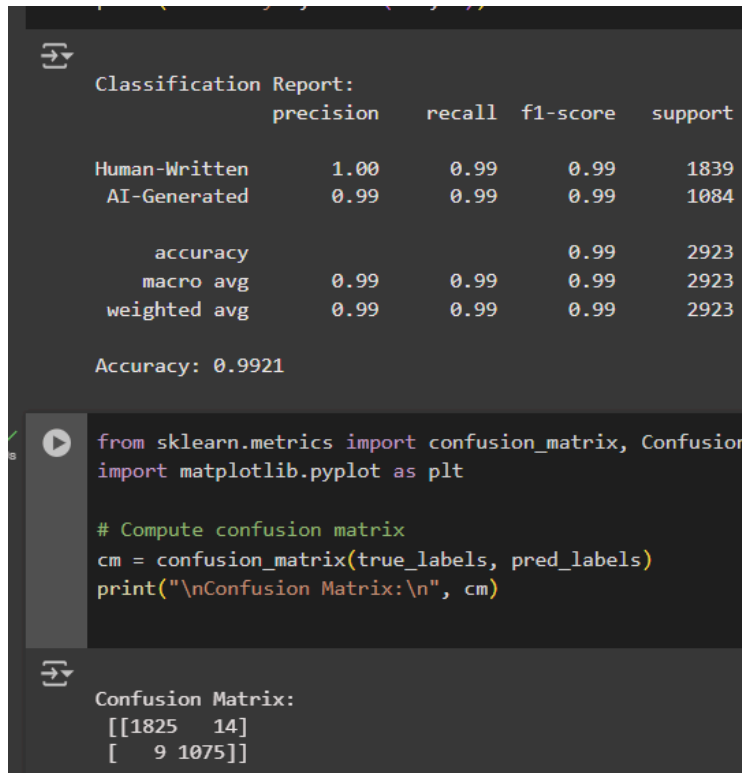
Accuracy = 99% time = 14m

```
✓ [42] # Fine-tune the model
14m trainer.train()
```



Epoch	Training Loss	Validation Loss	Accuracy	F1
1	0.034200	0.090688	0.980842	0.980916
2	0.007300	0.048632	0.992131	0.992135

```
TrainOutput(global_step=2924, training_loss=0.016128404560532654, metrics={'train
3.362, 'total_flos': 1549073759883264.0, 'train_loss': 0.016128404560532654, 'epc
```



The screenshot shows a Jupyter Notebook interface. The top part displays a 'Classification Report' with the following data:

	precision	recall	f1-score	support
Human-Written	1.00	0.99	0.99	1839
AI-Generated	0.99	0.99	0.99	1084
accuracy			0.99	2923
macro avg	0.99	0.99	0.99	2923
weighted avg	0.99	0.99	0.99	2923

Below the report, it states 'Accuracy: 0.9921'.

The bottom part of the notebook shows a code cell with the following Python code:

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

# Compute confusion matrix
cm = confusion_matrix(true_labels, pred_labels)
print("\nConfusion Matrix:\n", cm)
```

Below the code cell, the output shows the 'Confusion Matrix:' as:

```
[[1825  14]
 [   9 1075]]
```

Increasing the number of epochs means we let the model see the whole training data more times, so it has more chances to learn patterns and adjust its weights. This often improves accuracy if the model needs more training, but it can also lead to overfitting if we go too far. In our case, going from 1 to 2 epochs kept the accuracy at 99%, but the confusion matrix changed slightly, false positives went down from 35 to 14, and false negatives stayed almost the same at 9. This means the model's predictions improved a little in catching AI-generated texts. The training time increased from 7 minutes to 14 minutes because we doubled the number of passes through the data.

Compare Baseline and Fine-tuned Model performance:-

When we compare the baseline model to the fine-tuned Distilbert model, the fine-tuned model shows improvement in performance across most metrics (not by much but still better than the baseline model).

The baseline logistic regression model using tf-idf features reached an accuracy of 97.91%, with precision and recall both very high for each class (human: precision 0.99, recall 0.98; AI: precision 0.98, recall 0.96) and weighted F1 around 0.98. Its confusion matrix showed 21 false positives and 40 false negatives (which is way more than the fine-tuned model).

The fine-tuned Distilbert model reached 98.5% accuracy in its base configuration, which is about a 0.6 percentage point improvement over the baseline. Precision and recall were excellent for both classes (human: precision 1.00, recall 0.98; AI: precision 0.97, recall 0.99), and the weighted f1 score was also high at around 0.984. The confusion matrix for this model had 35 false positives and 9 false negatives, which is a trade-off compared to the baseline: false negatives dropped by 77% (40 → 9), but false positives increased (21 → 35). This shift means the fine-tuned model is far less likely to misclassify human-written text as AI-generated, but slightly more likely to misclassify AI-generated text as human-written.

Overall, the fine-tuned model performs better in accuracy, recall for the AI class, and especially in reducing false negatives, which is important if the goal is to reliably identify human-written content correctly. The trade-off is a small increase in false positives, which could be acceptable depending on the use case. The improvements are slight in percentage terms but significant in error reduction, especially for human text detection, where misclassifications dropped from 40 to just 9 cases.

Error Analysis:

We analysed a few examples of false positives and false negatives generated by the fine-tuned model.

For false positives (AI-generated texts classified as human-written), both examples are AI-written but contain a lot of grammatical mistakes, misspellings, and awkward phrasing that make them look like poorly written human essays. The model may have learned that such imperfections are more common in human writing, therefore it becomes biased with that notion and incorrectly labels them as human-written instead of AI-generated.

For false negatives (human-written texts classified as AI-generated), Even though there are still typos and grammatical in the human essays, they are well-structured, formal, and coherent, which makes them resemble the style of AI-generated text. In some cases, the vocabulary and sentence flow are relatively clean and consistent, which might trick the model into thinking they

were AI-generated. This suggests that our model may rely heavily on features like grammar, structure, and fluency, rather than deeper semantic or contextual hints.

Although in some examples,

Like

"In life, there is ups and downs. You can live your best life and still fail at times. Failure teaches most people lessons. People also don't care at all. Although, failure can also bring success, if you move smart. Yes, I agree with Prime Minister Churchill with this statement, it all adds up. This all depends on your level of enthusiasm. How you can handle failure and just grow, learn from mistakes. Failure can bring success. In life there is ups and downs. Nothing wrong with that at all."

Even though the human written content has typos and grammatical errors, the model still misclassified it as AI-generated. This might be due to the same reason as the baseline model, that the model is overfitting or that data is corrupted. The text has lots of odd spellings like "you'kg", "dogs't" etc. Subword tokenizers break these into unusual pieces, so if the model saw noisy AI samples in training with similar "glitchy" tokens, it can learn to relate that pattern with the AI-generated class. The length and essay-like outline with cliché statements can also contribute to the model leaning towards AI-generated text.

In short, the bias may be linked to the training data: if the AI-generated samples in training were mostly clean and grammatical, and human-written samples included many typos and informal expressions, the model would learn to use these as strong indicators which is not ideal in real life scenarios.

Ethical Considerations Analysis:

This project generates the model (baseline model) and fine-tunes a pre-trained model on AI vs Human dataset. The goal is to identify whether a text is AI-generated or Human written.

These models can be very useful in real life, but there are also ethical considerations. They can help teachers and schools check if homework or essays are written by students or by AI. They can be used by content moderators to stop fake news or spam generated by AI, and researchers or companies can use them to study how AI is being used and control harmful uses.

However, they can also cause harm. People who are still learning English might write in a way that looks similar to AI text (being proper in grammar and structure), so they could be wrongfully flagged as AI. People with disabilities who use writing aids or special tools might also be wrongly labeled as using AI. Even writings of creative writers, who use a very clean and formal style, might be mistaken for AI-generated text.

There can be bias in the dataset or the model. If most AI examples in the training data are perfect and clean, the model may think all perfect writing is AI. If most human-written examples in the data have more spelling mistakes or informal style, the model might learn that mistakes mean human and perfection means AI. Cultural and language differences can also cause problems, because writing styles from different regions might not be represented well, leading to more misclassifications.

These mistakes matter because people could be wrongfully accused of using AI when they did not use it. This could hurt students' grades, damage trust, or even cause problems at work. To avoid this, it is important to test the model on many kinds of writing and improve it so it treats everyone fairly. It is also important to ensure that the model is trained on uncorrupted data and does not overfit or gets biased.