BT5153 Group10 Detect AI-Generated Text

Chen Xiaofei (A0242321X)

Deng Shuanghong (A0280445E)

Xie Shiqi (A0280499M)

Liu Yuhang (A0280494X)

Zhang Hongyang (A0262805E)

Abstract

This project addresses the challenge of distinguishing between human-authored and AI-generated text using a meticulously curated dataset and comprehensive linguistic analysis. Traditional machine learning models, including logistic regression with TF-IDF and linguistic feature analysis, achieved high accuracy in text classification. Neural-based methods, such as DistilBERT, demonstrated outstanding performance, surpassing 93% accuracy. The study highlights the importance of responsible AI usage and transparent model interpretation for ensuring digital authenticity and trust. Future work includes refining models, expanding datasets, and extending real-time detection capabilities across media platforms.

1. Problem Statement

In today's digital landscape, the proliferation of large language models (LLMs) has revolutionised text generation capabilities, producing content that can closely mimic human writing. This technological advancement, while impressive, brings with it significant challenges to academic integrity, content authenticity, and digital trust. Misuses of this technology, such as the spread of fake news, impersonation, and academic dishonesty including plagiarism and AI-authored essays, pose serious threats to the credibility of educational entities, the authenticity of published content, and the reliability of online communication.

The core problem our project addresses is the need for robust mechanisms to discern AI-generated text from human-authored content. This capability is crucial for educational institutions, publishers, and content platforms that rely heavily on the authenticity and integrity of their text. By developing a model that can accurately identify AI-generated essays and other texts, we aim to bolster defences against the misuse of text generation technologies.

The importance of this project extends beyond mere content verification. It involves setting standards for ethical AI usage and establishing a framework that ensures AI tools enhance digital trust and content authenticity without compromising the credibility of digital content. By tackling these challenges, the project not only protects existing infrastructures but also pioneers responsible AI practices that could serve as benchmarks for future technological deployments. This endeavour is not just a response to a growing problem but a proactive step towards shaping the future of digital content creation and consumption in an AI-augmented world.

2. Dataset Description

The dataset at hand, as meticulously compiled and presented by Aaditya Bhat (2023), serves a critical function in the domain of computational linguistics and artificial intelligence. It has been curated for the express purpose of facilitating the training of machine learning models that discern between human-authored text and text generated by advanced generative models, specifically the GPT (Generative Pre-trained Transformer) variant known as 'Curie'.

2.1 Composition and Characteristics

This corpus encompasses introductory paragraphs drawn from Wikipedia for a sweeping range of 150,000 topics, alongside counterparts generated by the aforementioned GPT model. These introductory sections are constructed to adhere to a standardised format of 200 words, emulating the style typical of Wikipedia entries.

To initiate text generation, a fixed prompt is employed, incorporating the topic's title and the initial seven words of the genuine Wikipedia introduction. The GPT model tasked with text production operates under the following configuration parameters: model="text-curie-001", with a set temperature of 0.7 to modulate creativity, max_tokens capped at 300 to limit output length, top_p set to 1 to

allow full probability distribution, and penalties applied for frequency (frequency_penalty=0.4) and presence (presence_penalty=0.1), presumably to foster diversity and discourage repetition.

2.2 Data Schema

The dataset is systematically structured into discrete columns with precise data types and descriptors. The primary identifier id corresponds to a 64-bit integer, and accompanying attributes include the url of the source Wikipedia page, the title of the topic, and the textual content of both wiki_intro (sourced from Wikipedia) and generated_intro (produced by the GPT model). Quantitative attributes capture the length in words (title_len, wiki_intro_len, generated_intro_len) and tokens (prompt_tokens, generated_text_tokens), providing measures of text complexity and density.

3. Data Preprocessing

In the preprocessing phase of data preparation, the dataset `GPT-wiki-intro.csv` undergoes a series of transformations to render it suitable for the subsequent modelling tasks. The initial step involves reading the dataset into a pandas DataFrame. This is followed by the duplication of the dataset to establish distinct subsets for human-written and AI-generated texts, which are pivotal for training classification models to distinguish between these two types of text. The `human_df` subset is obtained by excluding the `generated_intro` column, signifying AI-crafted introductions, and consequently re-labeling the `wiki_intro` column as `text`. This subset is appended with a `label` column containing zeros, designating the human-authored nature of the text.

Conversely, the `ai_df` subset is derived by discarding the `wiki_intro` column, thereby removing the human-written content, and re-labelling the `generated_intro` as `text`. Analogously, a `label` column is appended, populated with ones to indicate AI-generated text. Subsequently, these subsets are amalgamated into a singular DataFrame, `combined_df`, which amalgamates human and AI-generated text samples, ensuring a balanced representation.

The consolidated dataset is then randomised using the `sample` method with a `random_state` for reproducibility, to mitigate any potential bias that could arise from the original ordering. The randomised dataset is finally exported to a CSV file, `combined_text_dataset.csv`, ensuring that the index is not included, thereby completing the data preprocessing stage which sets the foundation for the model to learn discriminative patterns inherent to each class of text.

4. Statistical Analysis

In order to reveal the differences between human-authored content and AI-generated content, we conducted a comprehensive statistical analysis of linguistic features, particularly lexical diversity, sentence complexity, readability scores, and frequency distribution of part of speech. By examining and visualising these features, we aim to uncover subtle and unique characteristics that help distinguish AI-generated content from human-written prose. The comprehensive plots are shown in Appendix A.

4.1 Lexical Diversity Analysis

The boxplot for lexical diversity shows that human-written text (label 0) tends to have a slightly higher lexical diversity than AI-generated text (label 1). This suggests that human writers typically use a broader vocabulary within their texts, while AI may be prone to reuse a more limited set of words. This characteristic can serve as a valuable marker for identifying AI-generated content.

4.2 Readability Score Analysis

Readability scores, as depicted in the box plot, reveal that human-written text has a broader interquartile range, implying a more variable readability level than AI-generated content. Interestingly, the median readability score for AI-generated text is higher, suggesting that such texts tend to be written in a way that is, on average, easier to read. This may reflect the optimisation algorithms in AI text generation that aim for clarity and understandability.

4.3 Average Sentence Length

The average sentence length in human-written texts displays significant variability, with some extreme outliers showing unusually long sentences, contributing to a more dynamic and varied reading experience. In contrast, AI-generated texts have a tighter distribution with fewer extreme values indicating a programmed adherence to syntactic construction rules that do not vary significantly.

4.4 Average Word Length

Both human and AI-generated texts exhibit similar distributions for average word length, with a slight tendency for AI-generated texts to have a less variable word length, which suggests a lack of nuanced word choice under certain contexts.

4.5 Stopwords Percentage

Regarding the percentage of stopwords the median and variability range is slightly higher on average in AI-generated texts, which indicates that AI-generated texts may use a higher proportion of common, less informative words than human-written texts, possibly due

to the nature of training algorithms that favour common language patterns for general applicability.

4.6 Frequency Distribution of Part of Speech (POS)

We calculated the frequency distribution of POS tags to gauge the syntactic variety and complexity within the text corpus. This distribution provides insightful metrics on the linguistic structure of the texts, highlighting the predominance of certain grammatical categories, such as nouns, verbs, adjectives, etc. which are essential for understanding the stylistic and functional aspects of the language. It extracted 45 POS tags individually, the summarised patterns are listed below:

4.6.1 Proper nouns (NNP) and nouns (NN)

Human-written contents show higher median and more variability in the use of proper nouns and common nouns, indicative of more detailed and context-specific content. The limited and repetitive use of nouns in AI-generated texts highlights its lack of detailed world knowledge.

4.6.2 Pronouns (PRP, PRP$)

Human-written contents have greater variability in personal and possessive pronouns, highlighting more complex narrative styles or dialogues. However AI-generated text lacks varied narrative perspectives.

4.6.3 Verbs (VBD, VBN, VBG, VB, VBP, VBZ)

Human-written contents use diverse patterns in verb forms, with noticeable outliers, reflecting a range of narrative tenses and a dynamic use of language. AI-generated texts generally consistently use fewer extremes.

4.6.4 Adjectives (JJ, JJR, JJS) and Adverbs (RB, RBR, RBS)

Humans use wider ranges and higher occurrences of adjectives and adverbs, especially with comparative and superlative forms, indicating richer descriptive content. While AI uses a narrower distribution of them, which indicates a less nuanced approach to description.

4.6.5 Function Words (IN, DT, CC) and Others (TO, MD)

Both show substantial use of function words, but humans exhibit greater variability, suggesting more complex sentence structures.

4.6.6 Special Characters and Symbols (Commas, Periods, Quotation Marks)

Human-written content contains variable and frequent punctuation usage, reflecting natural language flow and structure. AI use consistent but different patterns in punctuation usage to highlight differences in sentence structuring techniques.

4.6.7 Interjections (UH), Foreign Words (FW), and Symbols (SYM)

Humans use such elements to indicate stylistic flair or specific content needs, while AI uses them less due to a lack of spontaneity and contextual adaptability.

5. Traditional Machine-Learning Models

5.1 Linguistic Features + Logistic Regression

Logistic regression is a commonly-used machine learning model for binary classification, catering to our case where there exists two outcomes: human-written and AI-generated. Additionally, logistic regression provides a probabilistic nature, easy interpretability, high efficiency, strong robustness and flexibility. Therefore we trained 2 logistic regression models using different methodologies, such as TF-IDF and Linguistic Features Analysis.

5.1.1 TF-IDF

Firstly, we implemented TF-IDF to transform textual data into numerical format suitable for machine learning algorithms. This technique highlights the importance of a word based on the frequency it appears in a document balanced against its commonness across all documents. All texts were vectorised using TF-IDF to transform text data into numerical features, where each row represents a document and each column indicates a TF-IDF score.

We split the dataset into 80% for training and 20% for testing, fit the vectoriser on the training and test data and transformed them separately. Thereafter, we trained and tuned the logistic regression model on the vectorised training data. Essential hyperparameters of the logistic model, such as regularisation strength and solver, are adjusted based on cross-validation results to optimise the model's performance.

Lastly, we evaluated the model's performance using metrics such as accuracy, precision, recall, and F1-score, tested on the unseen test data.

	precision	recall	f1-score	support
0	0.96	0.96	0.96	29938
1	0.96	0.96	0.96	30062
accuracy			0.96	60000
macro avg	0.96	0.96	0.96	60000
weighted avg	0.96	0.96	0.96	60000

Figure 1. Model performance for logistic regression with TF-IDF

As shown in the above classification report, the model achieved promising results, with high values of 0.96 in precision and recall, indicating the model's effectiveness in distinguishing between human and AI texts. The analysis of the model coefficients revealed that certain terms significantly influence the classification, providing insights into features that are distinctly prevalent in human versus AI-generated texts.

5.1.2 Statistic-Based Detector - Linguistic Features Statistics

We have implemented one of the statistic-based detectors - linguistic feature statistic to our project. The extraction of linguistic features using NLTK and Textstat libraries offers a robust method for analysing textual data.

We initially used NLTK (Natural Language Toolkit) to tokenise text into words and sentences, tag parts of speech, and determine the frequency distribution of these tags, to facilitate detailed linguistic analysis. We utilised Textstat to calculate readability scores, providing insights into the complexity and understandability of the text.

Next, we defined various linguistic features such as lexical diversity, sentence complexity, readability, and frequency distributions of POS Tags. Thus we can gain deeper insights into the characteristics that distinguish human-written texts from those generated by AI. This approach not only aids in the classification of text origins but also enriches our understanding of language use in different contexts.

Thereafter, we loaded our datasets and extracted the linguistic features for all texts individually, and converted the list of dictionaries into a data frame. Extracting the linguistic features from the combined dataset rather than separating them by label beforehand achieved unbiased feature engineering, contained statistical consistency, and maintained scalable and simplified processing.

To train our model, we utilised features data frame as our model input, ‘label’ as the target variable, and split the dataset into 80% for training and 20% for testing. Prior to training the model, we scaled the feature data to ensure that no single feature dominates the model due to its scale, thus contributing to more stable and faster convergence during training. The scaler is fitted on the training data and subsequently used to transform both the training and testing datasets to maintain consistency in how data is treated across both phases.

We then trained and tuned the logistic regression model using scaled training data, allowing it to learn to differentiate between the classes based on the transformed features. Lastly, we generated the classification report to evaluate the model performance, shown below.

	precision	recall	f1-score	support
0	0.85	0.85	0.85	29938
1	0.85	0.85	0.85	30062
accuracy			0.85	60000
macro avg	0.85	0.85	0.85	60000
weighted avg	0.85	0.85	0.85	60000

Figure 2. Model performance for logistic regression with linguistic features

The logistic regression model demonstrated a consistent and commendable performance in classifying texts, achieving an overall accuracy of 85%. For both human-written (Class 0) and AI-generated texts (Class 1), the model recorded precision, recall, and F1-scores of 0.85. This indicates a high level of accuracy and balance in correctly identifying and classifying both types of texts. The macro and weighted averages for precision, recall, and F1-score uniformly stood at 0.85, reflecting the model's unbiased performance across both classes. This consistent efficacy across different volumes of class instances highlights the model’s robustness and its potential utility in applications requiring precise text source classification.

5.2 Linguistic Features-Based Classifiers using SVM and KNN

In addition to logistic regression, we extended the scope of feature-based classifiers to Support Vector Machines (SVM) and K-Nearest Neighbors (KNN) classifiers. We extracted a set of seven crucial linguistic features that are indicative of the stylistic, complexity, and sentiment characteristics of the texts. The features include:

- Capitalised Word Frequency: Measures the prevalence of capitalised words, reflecting the formal style often found in written texts.
- Stopword Frequency: Assesses the density of common words, which can indicate general text flow and readability.
- Quote Frequency: Quantifies the use of quotations, relevant to identifying narrative or reported speech.
- Punctuation Ratio: Evaluates the frequency of punctuation marks, which helps in understanding sentence structuring.
- Average Sentence Length: Provides insights into the complexity and readability of the text.
- Type-Token Ratio (TTR): A lexical diversity index measuring the ratio of unique words to the total number of words, showcases the complexity of the text.
- Sentiment Score: A psychological feature linked to sentiment analysis, derived using SentiWordNet (Baccianella, Esuli, & Sebastiani, 2010).

Prior to classification, the features are scaled to ensure that our SVM and KNN models function optimally without bias towards any disproportionately scaled feature. A standard scaler is implemented to normalise the data, providing each feature with equal initial weightage in the subsequent analysis.

For the classification task, SVM was chosen for its effectiveness in high-dimensional spaces, and KNN was utilised to exploit its capability of handling outliers and making decisions based on the majority vote from the nearest data points.

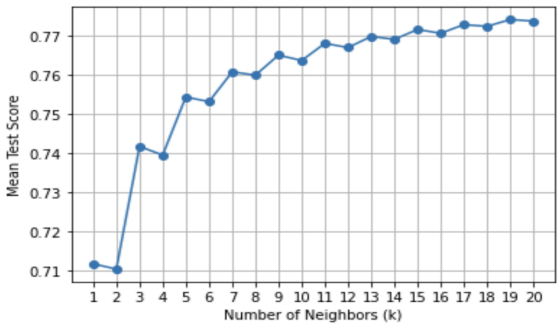


Figure 3. Model accuracy at different #. of neighbours

For the KNN classifier, the optimal number of neighbours (K) was determined through Grid Search, testing values from 1 to 20. The relationship between K values and the model's mean test score is depicted in Figure 3, which shows a clear trend: accuracy increases with higher K values. The graph highlights that K=19 offers the highest accuracy and was thus chosen for the final model applied to the test set.

Using K=19, our final KNN model exhibited robust performance metrics on our test data with accuracy: 77.93%, precision: 77.79%, recall: 78.17% and F1 Score: 77.98%. These results closely mirror the training outcomes, suggesting an absence of both overfitting and underfitting. This balance underscores the model's ability to generalise effectively across unseen data. In contrast, other studies have reported achieving up to 97% accuracy by extracting 21 textual features and utilising a KNN classifier (Aich, Bhattacharya, & Parde, 2022). This highlights the significance of selecting diverse and relevant textual features, which can significantly enhance the classifier's ability to distinguish between human-written and AI-generated texts.

When SVM is applied to scaled features, there is an improvement in model performance, with accuracy: 79.10%, precision: 78.91%, recall: 79.44% and F1 Score: 79.17%. This enhancement can largely be attributed to SVM's robustness against outliers, compared to KNN, which is more sensitive to local data distributions. Despite this improvement, the relatively modest gains suggest that the overall performance may still be constrained by the selection and quality of the extracted features. This indicates that optimising feature extraction could potentially lead to further gains in model accuracy and effectiveness.

6. Neural-Based Methods

6.1 LLM as Detectors

ChatGPT was utilised to classify texts generated by LLMs and those written by humans. However, the results were underwhelming, casting doubt on the reliability of using LLMs as detectors. The model often misclassified texts produced by LLMs as human-written. Intriguingly, the zero-shot setting outperformed both the one-shot and two-shot settings. Nevertheless, the accuracy rates for the one-shot and two-shot settings fell below 50% as shown in Appendix B. This underscores the decreasing dependability of using LLMs for direct detection of self-generated text, especially when contrasted with methods based on statistical and linguistic features.

6.2 DistilBERT

DistilBERT is a lightweight variant of the BERT model. It halves the original 12 transformer layers in the original BERT model, resulting in around a 40% decrease in size and 60% speed improvement, and at the same time retains 97% of BERT's performance on various NLP tasks. Hence, it is highly suitable for scenarios with limitations on computational resources and time. Compared to traditional NLP techniques, DistilBERT can capture the semantics and context of words, and handle out-of-vocabulary (OOV) words, but it requires a lot more computational resources compared to traditional machine learning models.

Prior to implementing the DistilBERT model, we tokenised each document in our dataset to visualise the distribution of token length across different documents in the corpus. We loaded the distilbert-base-uncased tokeniser from the Hugging Face library, as it is designed to handle inputs compatible with the DistilBERT architecture. We specified a maximum token length of 512 tokens, which is a common threshold in many NLP tasks, as it balances between capturing sufficient contextual information and maintaining manageable computation requirements. We visualised the token lengths for each document in the histogram in Figure 4. The bimodal distribution indicates that the most common token lengths fall within the range from 150 to 300 tokens, with only a few exceeding 512 tokens, assuring us that applying a maximum length of 512 tokens would not

lead to a significant loss of information. This step is crucial in preparation for the DistilBERT model, as understanding the distribution of token lengths helps us assess padding needs and evaluate truncation impact.

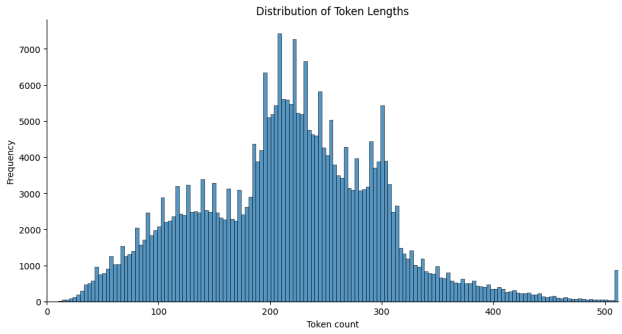


Figure 4. Distribution of Token Lengths

We split the dataset into training (60%), validation (20%) and testing (20%) datasets. A smaller proportion of the training set was chosen to speed up the training process.

We then constructed a DataLoader by defining a custom Dataset class to efficiently manage the loading and batching of our text data for input into the DistilBERT model, including using a tokeniser to convert each paragraph in the dataset while incorporating necessary parameters such as max_length to handle maximum token length, special tokens, padding and attention mask. We used a maximum token length of 512 and a batch size of 8. It is important to note that we did not remove stopwords during tokenisation because DistilBERT benefits from complete text data to understand the context and nuances of the language. This setup ensures that the input batches are uniformly structured, streamlining the subsequent learning process.

After creating the DataLoaders for training, validation and testing sets respectively, we moved on to the initialisation and training phase of the DistilBERT model using PyTorch Lightning, which simplifies the code from a typical PyTorch training loop.

First, we imported the pre-trained DistilBERT model specifically tailored for sequence classification tasks with AutoModelForSequenceClassification. We unfreezed the last two layers of the pre-trained model to adjust these parameters during the training process. This step allows the model training to focus on fine-tuning the most relevant parts of the model to our dataset and the binary classification task.

Then, we defined a custom PyTorch Lightning class called LLMDetector class, which specifies the training, validation, and testing steps. During each training batch, the model processes input tokens and attention masks, calculates the loss, predicts labels from logits, and logs relevant accuracy and loss metrics. Similar procedures are followed for validation and test batches where the model evaluates its performance on unseen data and logs the relevant metrics. We used an Adam optimiser, which is typical for training neural networks, and a linear scheduler to adjust the learning rate throughout the training process, which can help the model better converge by gradually decreasing the learning rate.

Next, a Trainer object of the PyTorch Lightning module is configured, specifying the number of epochs, device for training, and logging settings. This is used to fit the model to our dataset, where the model would pass the input data through the layers during forward propagation, calculate the loss function, and update the weights through backpropagation. After each training epoch, the model was evaluated on the separate validation set in the evaluation mode without backpropagation or parameter updates.

Finally, we plotted the history of validation accuracies in Figure 5, which was interpolated to omit minor decreases in accuracy. The graph shows that the accuracy continued to improve throughout the 10 epochs, among which 7 epochs achieved increases in validation accuracy. We selected the model with the highest validation accuracy as our "best model", which was then used to make predictions on the test dataset. The highest validation accuracy of 0.93155 was reached at the 9th epoch, and the test accuracy was 0.9325.

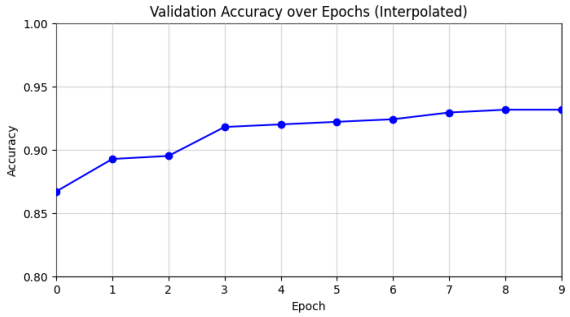


Figure 5. Validation Accuracy over Epochs

Overall, we reached a quite satisfying level of accuracy by fine-tuning the pre-trained DistilBERT model, with test and validation accuracy both exceeding 93%, indicating that our model generalises well to new, unseen data and demonstrates robust performance across different subsets of the dataset. This shows that by capturing the context of words and understanding the sequence of words, the DistilBERT model can distinguish the differences between human-written and AI-generated texts. At the same time, the downside is that, although a lightweight version of the BERT family, DistilBERT is still a transformer-based model that is relatively large and resource-intensive to train. For this specific task, the

training process took us more than 25 hours on an Apple M2 Pro 16-Core GPU.

7. Conclusion

In conclusion, our project has successfully developed and compared several models to detect AI-generated text, addressing crucial digital authenticity challenges.

Model	Accuracy	Advantages	Limitations
Logistic Regression with TF-IDF	96%	- Effective at handling sparse data - Interpretable model - Quick to train and predict	-Struggles with non-linear relationships - Relies on proper pre-processing to handle text data well
Logistic Regression with statistic-based detector	85%	- Incorporates linguistic features directly - Provides interpretable results - Good for small to medium datasets	- Requires careful feature selection - May not capture complex patterns in large datasets
KNN	78%	- Simple to implement and understand - No assumption about data distribution - Flexible to feature/distance choices	- Slow at making predictions in large datasets - Sensitive to noisy data and irrelevant features
SVM	79%	- Effective in high-dimensional spaces - Works well with clear margin of separation - Versatile with different kernels	- Requires full data loading for training - Intensive memory usage - Difficult to choose appropriate kernel
DistilBERT	93%	-State-of-the-art performance for text data - Understands context and semantics - Reduced model size from BERT	- Still requires significant computational resources- May need fine-tuning for specific tasks

Figure 6. Comparison of different models

Based on the performance metrics we have, the Logistic Regression model with implementation of TF-IDF exhibits the highest level of accuracy among the evaluated models, achieving a precision, recall, and F1 score of 96%. This indicates a highly effective model in distinguishing between human-written and AI-generated texts. It is effective at handling sparse data and provides a model that is both interpretable and quick to train and predict. However, it may struggle with non-linear relationships and heavily relies on proper pre-processing to handle text data effectively. Additionally, the Logistic Regression model with a statistic-based detector demonstrated a consistent and commendable performance in classifying texts, achieving an overall accuracy of 85%. The advantage of this model leverages the direct incorporation of linguistic features, enhancing interpretability, but it requires careful feature selection and may not capture complex patterns in large datasets. In comparison, both the SVM and KNN models show somewhat lower performance, with precision and recall rates around 78%, and F1 scores of 79% and 78% respectively. These models, while useful, appear less consistent in correctly identifying the text origins compared to Logistic Regression. KNN excels in scenarios where the decision boundary is not linear. However, KNN can be computationally expensive for large datasets and is sensitive to noisy data and irrelevant features. SVM is effective in high-dimensional spaces and can handle complex datasets with a clear margin of separation, supported by its versatility with different kernels. Yet, SVMs suffer from high memory usage and the computational burden increases with data size, and selecting the appropriate kernel can be challenging. DistilBERT, a more complex neural network model, also shows strong performance with all metrics at 93%, suggesting that despite the higher computational demands, it is highly effective for tasks requiring a nuanced understanding of the text. This demonstrates DistilBERT's capability to capture the complexities of language that differentiate human and AI-generated content. Despite these advantages, DistilBERT still demands considerable computational resources and may require specific fine-tuning for particular tasks, which

could be a limitation in resource-constrained environments. Thus, while Logistic Regression leads in raw performance metrics, DistilBERT offers a robust alternative when higher model complexity and deeper linguistic analysis are required.

8. Future Work

Looking forward, there is still space to enhance the precision and applicability of AI text detection models. Future work will focus on refining model structures and expanding training datasets to cover more nuanced linguistic features. Moreover, extending these models to detect AI-generated content in real-time across different media remains a critical need, particularly for platforms where immediate content verification is crucial. We also advocate for continued research into the ethical use of AI, ensuring that advancements in detection technology are used responsibly and transparently.

References

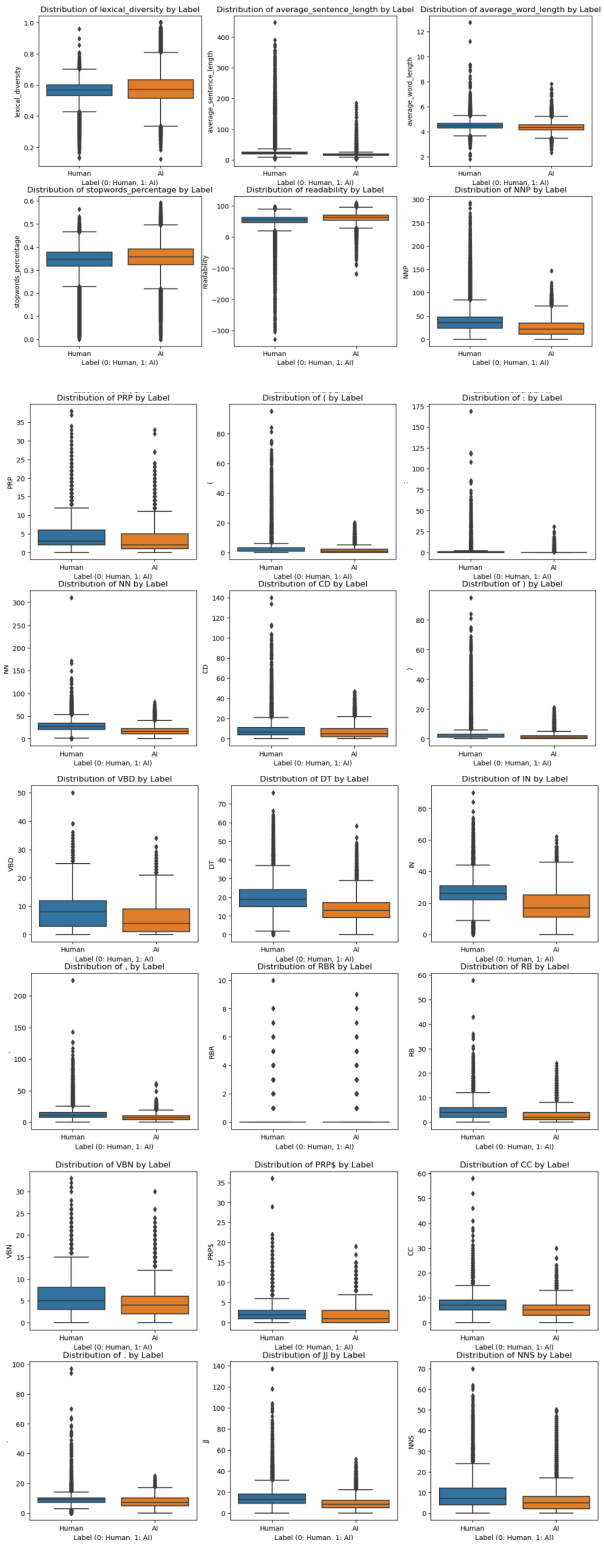
Bhat, A. (2023). GPT-wiki-intro (Revision 0e458f5). HuggingFace. https://huggingface.co/datasets/aadityaubhat/GPT-wiki-intro.https://doi.org/10.57967/hf/0326

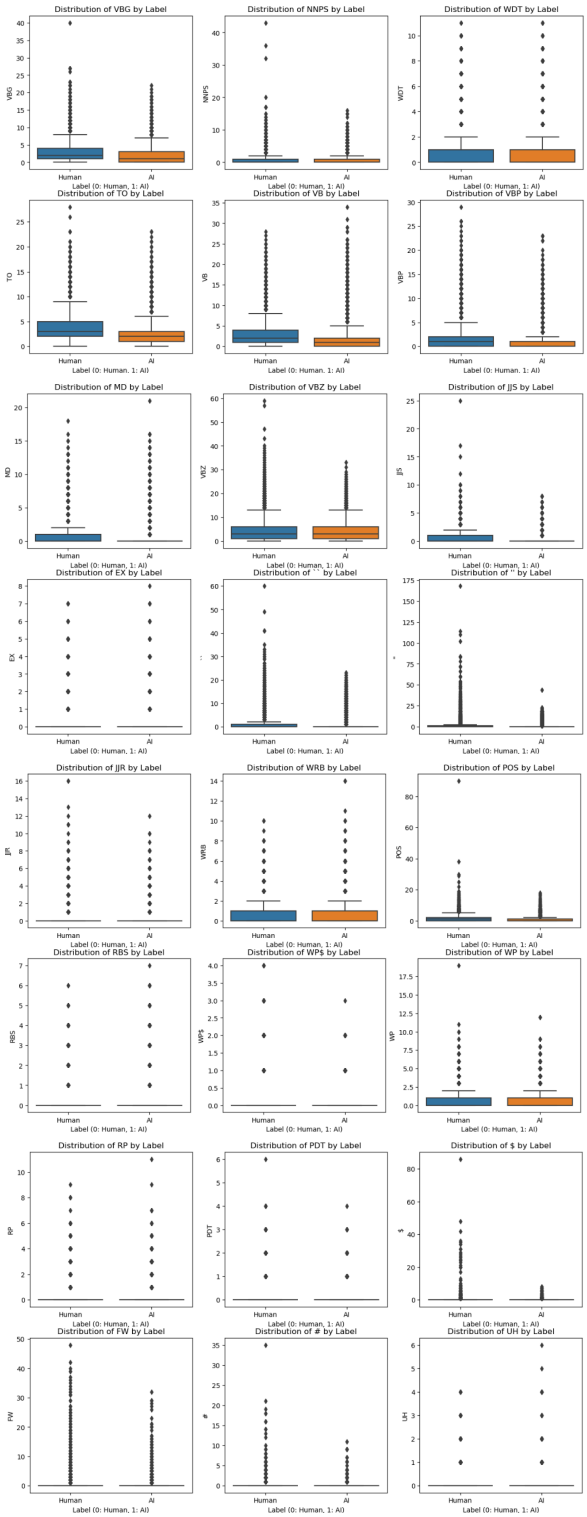
Baccianella, S., Esuli, A., & Sebastiani, F. (2010). SentiWordNet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In Proceedings of the International Conference on Language Resources and Evaluation, LREC 2010, 17-23 May 2010, Valletta, Malta (pp. 2200-2204). European Language Resources Association (ELRA).

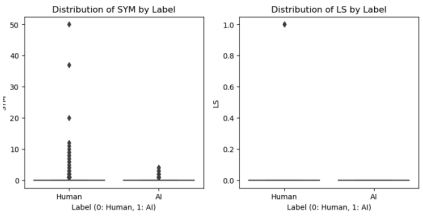
Aich, A., Bhattacharya, S., & Parde, N. (2022). Demystifying neural fake news via linguistic feature-based interpretation. In Proceedings of the 29th International Conference on Computational Linguistics, COLING 2022, October 12-17, 2022, Gyeongju, Republic of Korea (pp. 6586-6599). International Committee on Computational Linguistics.

Appendix

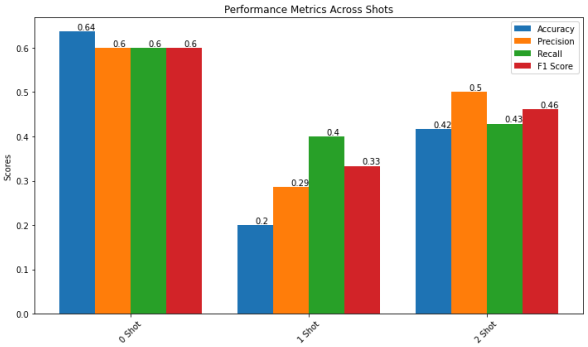
Appendix A: Plots of Linguistic Features







Appendix B: Results using ChatGPT as Detector



Appendix C: Project Repository

GitHub Repository "BT5153-Detect-AI-Generated-Text": https://github.com/Rihond/BT5153-Detect-AI-Generated-Text