

**Detecting AI-Generated Text in Long-Form Content Using NLP Techniques**

***A Study on Enhancing Academic Integrity and Improving AI Training***

Michael Shpyl

BSc (Hons) in Computing Science

2024/2025

**Detecting AI-Generated Text in Long-Form Content Using NLP Techniques**

***A Study on Enhancing Academic Integrity and Improving AI Training***

Author: Mykhailo Shpyl

Supervised by: Kevin Meehan, Mandy Douglas

A thesis submitted in partial fulfilment of the requirements for “Research in Computing with Emerging Technologies” and “Project Development”

BSc (Hons) in Computing Science

Submitted to Atlantic Technological University

*Arna chur isteach chuig Ollscoil Teicneolaiochta an Atlantaigh* October 2024

# 

# Declaration

I hereby certify that the material, which l now submit for assessment on the programmes of study leading to the award of BSc (Hons) in Computing Science , is entirely my own work and has not been taken from the work of others except to the extent that such work has been cited and acknowledged within the text of my own work. No portion of the work contained in this thesis has been submitted in support of an application for another degree or qualification to this or any other institution. I understand that it is my responsibility to ensure that I have adhered to ATU’s rules and regulations.

I hereby certify that the material on which I have relied on for the purpose of my assessment is not deemed as personal data under the GDPR Regulations. Personal data is any data from living people that can be identified. Any personal data used for the purpose of my assessment has been psudonymised and the data set and identifiers are not held by ATU. Alternatively, personal data has been anonymised in line with the Data Protection Commissioners Guidelines on Anonymisation.

I consent that my work will be held for the purposes of education assistance to future students and will be shared on the ATU Donegal (Computing) website (www.lyitcomputing.com) and Research THEA website (https://research.thea.ie/). I understand that documents once uploaded onto the website can be viewed throughout the world and not just in the Ireland. Consent can be withdrawn for the publishing of material online by emailing Thomas Dowling; Head of Department at thomas.dowling@lyit.ie to remove items from the ATU Donegal Computing website and by email emailing Denise McCaul; Systems Librarian at denise.mccaul@lyit.ie to remove items from the Research THEA website. Material will continue to appear in printed formats once published and as websites are public medium, ATU cannot guarantee that the material has not been saved or downloaded.

Signature of Candidate: Mykhailo Shpyl Date: 25/04/2025

# Acknowledgements

I would like to thank…

Kevin Meehan

Mandy Douglas

Martin Robinson

# Acronyms

| **Acronym** | **Definition** |
| --- | --- |
| AI | Artificial Intelligence |
| NLP | Natural Language Processing |
| LLM | Large Language Model |
| GenAI | Generative Artificial Intelligence |
| GPT | Generative Pre-trained Transformer |
| BERT | Bidirectional Encoder Representations from Transformers |
| RoBERTa | Robustly Optimized BERT Approach (transformer-based model) |
| LIME | Local Interpretable Model-Agnostic Explanations |
| XAI | Explainable Artificial Intelligence |
| ROC | Receiver Operating Characteristic (curve) |
| PR | Precision-Recall (curve) |
| API | Application Programming Interface |

# Table of Contents

[Declaration 3](#_Toc196493040)

[Acknowledgements 5](#_Toc196493041)

[Acronyms 6](#_Toc196493042)

[Table of Contents 7](#_Toc196493043)

[Table of Figures 10](#_Toc196493044)

[Table of Tables 11](#_Toc196493045)

[Chapter 1: Introduction 1](#_Toc196493046)

[1.1 Background. 2](#_Toc196493047)

[1.2 Problem Statement. 2](#_Toc196493048)

[1.3 Research Questions and Objectives. 3](#_Toc196493049)

[1.3.1 This research is guided by several key questions: 3](#_Toc196493050)

[1.3.2 To address these questions, the objectives of this project are: 4](#_Toc196493051)

[1.4 Significance of the Research. 4](#_Toc196493052)

[1.5 Scope and Limitations. 4](#_Toc196493053)

[2.1 Introduction 5](#_Toc196493054)

[2.2 Detection Tools and Methods 6](#_Toc196493055)

[2.2.1 Statistical and Linguistic Approaches 6](#_Toc196493056)

[2.2.2 Watermarking and Provenance Techniques 7](#_Toc196493057)

[2.2.3 Hybrid Models and Machine Learning-Based Detection 7](#_Toc196493058)

[2.2.4 Critical Considerations and Limitations 7](#_Toc196493059)

[2.3 Natural Language Processing Techniques for Detection 8](#_Toc196493060)

[2.3.1 Foundational Preprocessing Techniques 8](#_Toc196493061)

[2.3.2 Advanced Semantic and Contextual Techniques 9](#_Toc196493062)

[2.4 Text Preprocessing and Feature Engineering 10](#_Toc196493063)

[2.4.1 Foundational Preprocessing Techniques 10](#_Toc196493064)

[2.4.2 Feature Engineering for AI Detection 11](#_Toc196493065)

[2.5 NLP Models for AI-Generated Text Detection 11](#_Toc196493066)

[2.5.1 Supervised Learning Models 12](#_Toc196493067)

[2.5.2 Unsupervised Learning Models 14](#_Toc196493068)

[2.5.3 Semi-Supervised and Hybrid Approaches 14](#_Toc196493069)

[2.5.4 Transformer-Based Architectures 15](#_Toc196493070)

[2.5.5 Model Comparisons 16](#_Toc196493071)

[2.6 Machine Learning (ML) in AI-Generated Text Detection 17](#_Toc196493072)

[2.6.1 Pattern Recognition in AI Detection 18](#_Toc196493073)

[2.6.2 Handling Labeled and Unlabeled Data 18](#_Toc196493074)

[2.6.3 Model Adaptability Over Time 18](#_Toc196493075)

[2.6.4 Challenges in Machine Learning for AI Detection 19](#_Toc196493076)

[2.6.5 Use of Ensemble Methods 19](#_Toc196493077)

[2.6.6 Conclusion 20](#_Toc196493078)

[2.7 Classification Algorithms and Ensemble Methods for AI-Generated Text Detection 20](#_Toc196493079)

[2.7.1 Classification Algorithms 21](#_Toc196493080)

[2.7.2 Ensemble Methods 21](#_Toc196493081)

[2.7.3 Benefits of Ensemble Methods in AI Detection 22](#_Toc196493082)

[2.7.4 Challenges with Ensemble Methods 22](#_Toc196493083)

[2.7.5 Conclusion 23](#_Toc196493084)

[2.8 Length-Sensitive Detection Techniques 23](#_Toc196493085)

[2.8.1 Handling Long-Form Content 23](#_Toc196493086)

[2.8.2 Adversarial Robustness in Long Texts 24](#_Toc196493087)

[2.8.3 Dealing with Paraphrasing in Long-Form Text 24](#_Toc196493088)

[2.8.4 Hybrid Models for Long Content 24](#_Toc196493089)

[2.8.5 Scalability Challenges 25](#_Toc196493090)

[2.8.6 Application in Journalism and Academia 25](#_Toc196493091)

[2.8.7 Conclusion 25](#_Toc196493092)

[2.9 Academic Integrity and Ethical Concerns 25](#_Toc196493093)

[2.9.1 Impact on Academic Integrity 26](#_Toc196493094)

[2.9.2 Role of Detection Tools 26](#_Toc196493095)

[2.9.3 Educational Policies and Guidelines 27](#_Toc196493096)

[2.9.4 Training and Awareness 27](#_Toc196493097)

[2.9.5 Societal Implications and Journalistic Perspective 27](#_Toc196493098)

[2.9.6 Balancing Innovation and Integrity 28](#_Toc196493099)

[2.9.7 Conclusion 28](#_Toc196493100)

[2.10 Filtering AI-Generated Content 29](#_Toc196493101)

[2.10.1 Importance and Risks of Recursive Training 29](#_Toc196493102)

[2.10.2 Methods for Filtering AI-Generated Content 29](#_Toc196493103)

[2.11.3 Mitigating Model Collapse 29](#_Toc196493104)

[2.11.4 Ethical and Practical Considerations 30](#_Toc196493105)

[2.11.5 Conclusion 30](#_Toc196493106)

[3. Design Chapter 30](#_Toc196493107)

[3.1 Introduction 30](#_Toc196493108)

[3.2 Data Acquisition and Preprocessing 31](#_Toc196493109)

[3.2.1 Dataset Choice and Citation 31](#_Toc196493110)

[3.7 Functional & Non-Functional Requirements 40](#_Toc196493111)

[3.7.1 Functional Requirements 40](#_Toc196493112)

[3.7.2 Non-Functional Requirements 41](#_Toc196493113)

[3.7 Summary 41](#_Toc196493114)

[Chapter 4: Implementation 42](#_Toc196493115)

[4.3 Choice of Models: BERT, RoBERTa, and Longformer. 51](#_Toc196493116)

[4.4 Model Training and Optimization. 55](#_Toc196493117)

[4.5 Interpretability with LIME. 58](#_Toc196493118)

[4.6 User Interface Development: Browser Extension and Dashboard. 61](#_Toc196493119)

[Chapter 5: Testing and Results 67](#_Toc196493120)

[5.1 Evaluation Methodology. 67](#_Toc196493121)

[5.2 Performance Metrics on the Test Set. 68](#_Toc196493122)

[5.3 Confusion Matrix and Detailed Analysis. 70](#_Toc196493123)

[5.4 Precision-Recall and ROC Curves. 71](#_Toc196493124)

[5.5 Additional Experiments and Visualizations. 72](#_Toc196493125)

[5.6 Longitudinal Trend Analysis (2015–2025). 76](#_Toc196493126)

[5.7 Scalability and Computational Considerations. 78](#_Toc196493127)

[5.8 Summary of Results. 79](#_Toc196493128)

[Chapter 6: Conclusion 80](#_Toc196493129)

[6.1 Answering the Research Questions. 80](#_Toc196493130)

[6.2 Achievements of the Project. 85](#_Toc196493131)

[6.3 Limitations and Future Work. 87](#_Toc196493132)

[6.4 Final Recommendations. 88](#_Toc196493133)

[6.5 Concluding Remarks. 90](#_Toc196493134)

[References i](#_Toc196493135)

# Table of Figures

# Table of Tables

# Chapter 1: Introduction

Artificial Intelligence (AI), particularly in the field of Natural Language Processing (NLP)—a branch of AI focused on enabling machines to interpret, understand, and generate human language—has experienced rapid advancements in recent years. A notable development within NLP is the rise of Generative AI (GenAI) models, sophisticated algorithms capable of producing highly coherent text closely resembling human writing. Among the most influential GenAI models are GPT-3 and GPT-4, which have demonstrated remarkable capabilities, from drafting articles to assisting in academic research (Brown *et al.*, 2020; OpenAI, 2023; Jurafsky & Martin, 2022).

However, as beneficial as these technologies can be, their widespread use poses significant challenges in journalism and academia. In journalism, AI-generated articles may appear credible but can inadvertently spread misinformation, bias, or incomplete narratives. If such content remains undetected, it can erode public trust, as readers rely heavily on the integrity and accuracy of news sources (Zellers *et al.*, 2019; Wu *et al.*, 2024; Reuters Institute, n.d.; Tow Center, n.d.; JournalismAI, n.d.; Nieman Lab, n.d.; The Trust Project, n.d.). In education, students can now easily access GenAI tools (e.g., ChatGPT) to help with academic assignments. While these tools can aid learning by providing hints or explanations, overreliance on AI-generated text risks unintentional plagiarism, undermining the authenticity of student work (Guillén-Yparrea & Hernández-Rodríguez, 2024; Eke, 2023; Uzun, 2023; ICAI, n.d.; Turnitin, n.d.; UNESCO, n.d.).

These challenges underline the importance of developing robust methods to detect AI-generated content. Ensuring that AI-driven misinformation does not compromise journalistic standards is vital for maintaining credibility and public trust. Similarly, preventing undetected AI-generated plagiarism in academia preserves academic integrity and the genuine value of student effort. Moreover, filtering AI-generated text from training datasets can improve future AI models by ensuring they learn from authentically human-authored materials, thereby enhancing model quality and contributing to more trustworthy and transparent AI systems (Prova, 2024; Chen *et al.*, 2024; Gebru *et al.*, 2021; Google Research, n.d.).

Addressing these complexities requires a detection tool capable of identifying AI-generated content and justifying its findings. Trust in such a tool is essential, as errors—mislabeling human-written work as AI-generated or failing to detect AI-driven misinformation—can be costly. A robust solution must provide clear, provable evidence to support its verdicts, ensuring educators, journalists, and other stakeholders understand and trust the detection outcomes (European Commission’s High-Level Expert Group on AI, n.d.; DARPA’s XAI, n.d.; CAI, n.d.; C2PA, n.d.; Partnership on AI, n.d.). This need for explainability aligns with broader efforts in Explainable AI (XAI) to make AI decisions more transparent and accountable.

* 1. Background.

The growing sophistication of Large Language Models (LLMs) has complicated the task of distinguishing AI-generated text from human-written content. Early methods often relied on simple lexical or statistical cues, but as GenAI models evolved, their outputs became more nuanced, rendering surface-level analysis insufficient. The complexity of AI writing now demands tools that can handle subtleties in language use, style, and structure across long-form documents (Gehrmann *et al.*, 2019; Mitchell *et al.*, 2023; Jurafsky & Martin, 2022; Vaswani *et al.*, 2017). In particular, AI-generated text can mimic human idiosyncrasies, making detection very challenging.

While GenAI can assist learning by offering clarifications in academia, it also makes it more challenging for educators to verify that submitted work reflects genuine student effort. Although guidelines and discussions on responsible AI use in education are emerging, reliable detection methods remain paramount (ICAI, n.d.; Turnitin, n.d.; UNESCO, n.d.; EDUCAUSE, n.d.). Similarly, in journalism, editors must ensure that AI-generated material does not introduce misinformation or bias, thus preserving editorial integrity and public trust (Zagorulko, 2023; Zellers *et al.*, 2020; Reuters Institute, n.d.). These concerns set the stage for developing a new detection approach focused on long-form content.

## 1.2 Problem Statement.

Existing detection methods, from basic keyword searches to advanced machine learning models, often struggle against the complexity of AI-generated texts. Techniques like term-frequency inverse document frequency (TF-IDF), once useful for straightforward plagiarism detection, falter when confronted with AI-generated content that closely mimics human linguistic patterns (Induri *et al.*, 2024). Even state-of-the-art transformer-based detectors have difficulty identifying subtly paraphrased AI-generated text designed to evade detection (Sadasivan *et al.*, 2023; Prova, 2024).

Moreover, detection systems must move beyond mere identification; they need to provide transparent evidence to justify their decisions. Without trustworthy explanations, educators may hesitate to challenge suspicious student work, and journalists may be reluctant to act on automated flags for fear of false accusations (OpenAI, Hendrik Kirchner *et al.*, 2023; AI2, n.d.). Reducing such errors is essential: mislabelling human text as AI-generated undermines confidence in the tool while failing to detect AI-driven misinformation can have serious societal consequences. There is also a broader concern that AI models trained on AI-generated data could reinforce each other’s imperfections. Filtering AI-generated content from training datasets could help future models rely more on genuine human writing, reducing biases and feedback loops (Chen *et al.*, 2024; Stray, 2023; Gebru *et al.*, 2021). By implementing approaches that are both effective and explainable, current detection capabilities can be strengthened and more trustworthy AI ecosystems can be fostered.

## **1.3 Research Questions and Objectives**.

### 1.3.1 This research is guided by several key questions:

* *What NLP techniques can be employed to develop a reliable tool for detecting AI-generated text, especially in longer pieces such as news articles and academic papers?*
* *How can educators and journalistic institutions utilize the tool to identify AI-generated content, maintaining credibility, academic integrity, and public trust?*
* *What are the limitations of existing AI-detection tools, and how can a new tool overcome these challenges to ensure user-friendliness, accuracy, and the ability to provide justifiable, provable evidence for its decisions?*
* *How can filtering AI-generated content during the development and training of future AI models improve the reliability of detection systems and enhance model quality?*

### 1.3.2 To address these questions, the objectives of this project are:

* To create a detection tool that effectively identifies AI-generated text in long-form content, focusing on news articles and academic papers.
* To implement NLP-based methods that adapt to sophisticated AI-generated outputs and improve detection robustness over baseline techniques.
* To develop an interface and workflow enabling educators and journalists to use the tool confidently, supported by transparent explanations and concrete evidence of AI authorship.
* To investigate how excluding AI-generated content from training datasets affects detection performance and whether this practice fosters more reliable, trustworthy AI models.
* To establish techniques for a system that not only flags AI-generated content but also justifies its conclusions with evidence, thereby reducing errors and enhancing trust.

## 1.4 Significance of the Research.

This research addresses urgent needs in both academic and journalistic contexts. Enabling reliable detection of AI-generated material helps educational institutions uphold academic standards and encourages responsible AI use among students (Eke, 2023; Uzun, 2023). In journalism, a dependable detector supports credibility by ensuring that information presented to the public is accurate and not secretly authored by AI (Reuters Institute, n.d.; Tow Center, n.d.; The Trust Project, n.d.). Additionally, by examining how filtering AI-generated content from training sets can improve future AI models, this work contributes to long-term stability and trustworthiness in AI systems. The broader impact involves promoting the responsible integration of AI into content creation, thereby safeguarding informational quality and maintaining trust in educational and journalistic institutions (UNESCO, n.d.; OECD AI Policy Observatory, n.d.).

## 1.5 Scope and Limitations.

This study focuses on English-language long-form texts, including news articles and academic documents. A custom dataset was constructed for this project, mixing human-written content with AI-generated samples created via advanced language models (rather than relying on any pre-labelled corpus of AI text). While every effort is made to detect even subtle AI paraphrasing, it is acknowledged that some AI-generated content may still evade detection due to the models’ ever-improving ability to mimic human style (Sadasivan *et al.*, 2023).

This research concentrates on text-based detection and does not cover other media types (e.g., image or video deepfakes). Future work may extend these techniques to multiple modalities and languages, further enhancing global integrity and trust in AI-assisted content creation. The scope is also constrained by practical considerations: computational resources limit the size of models and datasets that can be used, and the most advanced proprietary AI (such as GPT-4) was not utilized extensively due to cost and access constraints. These limitations inform certain trade-offs in the design and implementation of the detection tool, as discussed in later chapters.

## 2.1 Introduction

The rapid growth of Artificial Intelligence (AI) in recent years has significantly influenced numerous domains, including journalism and academia, altering how content is produced and shared (Brynjolfsson & McAfee, 2017; Russell & Norvig, 2020). AI-driven text generation models have become increasingly sophisticated, enabling the creation of highly convincing human-like text. One prominent family of these models is the Generative Pre-trained Transformer (GPT) series, which utilizes advanced architectures to generate coherent and contextually relevant prompt responses (Brown et al., 2020). Similarly, techniques from Natural Language Processing (NLP)—an interdisciplinary field at the intersection of computer science, linguistics, and AI—provide the computational methods for processing, analyzing, and generating human language.

While these advances offer new opportunities for automating content creation, they pose serious challenges. Within journalism, the rise of AI-generated news articles complicates verifying sources and authenticity, potentially affecting public trust and the media’s credibility. Academic environments face related ethical and practical concerns, such as students using AI-generated text to complete assignments, thereby undermining academic integrity (Eke, 2023; Uzun, 2023). Furthermore, given the rapid development in AI text generation, conventional plagiarism detection tools often struggle to identify sophisticated machine-generated outputs.

Researchers and practitioners are exploring methods to reliably distinguish between human-written and AI-generated text to address these issues. While a range of approaches exist—from statistical analysis to advanced NLP techniques—none are entirely foolproof. The current literature increasingly focuses on improving the effectiveness of these detection strategies and understanding their limitations. This includes evaluating the accuracy and scalability of detection methods and critically examining their ethical implications, maintenance requirements, and sensitivity to evolving AI models.

## 2.2 Detection Tools and Methods

Efforts to identify AI-generated text have led to a variety of detection methods. These methods generally fall into three main categories: (1) statistical and linguistic analysis, (2) watermarking and related provenance techniques, and (3) hybrid approaches that combine traditional feature-based analysis with advanced machine learning models. Although these tools have shown promise, each comes with its own limitations, especially as AI-generated text grows increasingly sophisticated and models are continuously refined.

### 2.2.1 Statistical and Linguistic Approaches

Early attempts at detecting AI-generated text often relied on statistical measures of language patterns. Metrics such as perplexity—which assesses how predictable a piece of text is—have been commonly used. AI-generated text is frequently more predictable than human writing, resulting in lower perplexity scores (Li et al., 2024; Mitchell et al., 2023). Other linguistic features, including word distribution, sentence structure, and vocabulary richness, can also be informative. However, a key limitation is that as AI models improve, their outputs become more varied and less distinguishable from human text on these simple metrics alone. This challenges purely statistical methods and calls for more nuanced approaches that consider semantic coherence and contextual appropriateness.

### 2.2.2 Watermarking and Provenance Techniques

Beyond statistical measures, some approaches involve embedding hidden markers—or “watermarks”—into AI-generated text. For instance, synthetically inserted linguistic patterns can serve as an invisible signature that a detection tool can later verify, effectively indicating the text’s origin (Dathathri et al., 2024). While watermarking can be effective for texts generated from known models under controlled conditions, it is less applicable to scenarios where the detector does not have control over the text production process. Additionally, sophisticated paraphrasing or adversarial modifications can potentially remove or obscure these watermarks, limiting their reliability in open-world conditions.

### 2.2.3 Hybrid Models and Machine Learning-Based Detection

More recent research has focused on hybrid detection models that combine traditional linguistic features—such as term frequency-inverse document frequency (TF-IDF)—with machine learning classifiers or transformer-based architectures (Zhang et al., 2024). By using features derived from NLP techniques alongside advanced classification algorithms (e.g., random forests, gradient boosting methods, and transformer-based encoders), these hybrid systems are better equipped to capture subtle patterns indicative of AI-generated content. Although hybrid methods can achieve higher accuracy, they may require substantial computational resources and large amounts of training data. Moreover, as AI-generation models evolve, detection models need regular updating and retraining to maintain performance, highlighting a practical challenge in sustaining long-term reliability.

### 2.2.4 Critical Considerations and Limitations

While these detection methods are integral to addressing the authenticity issues posed by AI-generated text, it is important to evaluate their limitations critically. Many detection tools lack cross-model generalization, performing well against the specific type of AI model they were trained on but failing to detect outputs from different architectures. Furthermore, certain techniques may inadvertently flag genuine human writing as machine-generated, raising concerns about false positives and the consequences of misclassification—particularly in high-stakes contexts like academic assessment or journalistic reporting. This underscores the need for further research, more diverse datasets, and continuous refinement of detection algorithms.

In summary, detection methods have evolved from simple statistical measures to more complex, hybrid strategies. Although promising, each approach faces challenges—ranging from the ease of bypassing watermarking to the complexity and resource intensity of advanced machine learning solutions. Critical engagement with these methods is necessary to ensure that future detection tools are both accurate and robust against the ever-changing landscape of AI-generated content.

## 2.3 Natural Language Processing Techniques for Detection

Natural Language Processing (NLP) underpins the methodologies used to detect AI-generated text by providing tools and models that help computers interpret and analyze human language. The core NLP techniques involved in detection tasks often include foundational text processing steps (e.g., tokenization, normalization) followed by higher-level lexical and semantic analyses. These techniques are essential for breaking down text into machine-readable units, identifying stylistic and lexical irregularities, and leveraging advanced models that capture syntactic and semantic nuances.

### 2.3.1 Foundational Preprocessing Techniques

#### 2.3.1.1 Tokenization

Tokenization is a fundamental step in NLP, involving the segmentation of a text into smaller units known as tokens—typically words or word-like entities. By decomposing input text at a granular level, detection algorithms can analyze patterns in individual tokens, making it easier to identify repetitive sequences or unusual word choices characteristic of AI-generated output. For instance, some AI models may overuse high-frequency tokens or maintain overly consistent sentence lengths, both of which can be detected through token-level analysis. Research indicates that improved tokenization methods can enhance cross-lingual transfer and parsing accuracy, ultimately benefiting downstream tasks like AI-generated text detection (Che et al., 2018; Manning et al., 2008).

#### 2.3.1.2 Normalization and Lexical Analysis

Normalization processes, such as converting text to lowercase, removing punctuation, or handling morphological variants, ensure that different forms of the same word are treated uniformly (Manning et al., 2008). In conjunction with normalization, lexical analysis focuses on word frequency, vocabulary diversity, and the distribution of specific terms. The AI-generated text might exhibit reduced lexical richness or favour certain high-probability terms, contrasting with the more varied and context-dependent choices made by human writers. Multiple studies have shown that features derived from lexical analysis can improve detection rates, although their effectiveness may decrease as AI models become more adept at mimicking human lexical choices (Gehrmann et al., 2019; Ippolito et al., 2020).

## 2.3.2 Advanced Semantic and Contextual Techniques

#### 2.3.2.1 Transformer-Based Models

Transformer-based models, such as Bidirectional Encoder Representations from Transformers (BERT), have revolutionized NLP by capturing the subtle nuances of language better than earlier architectures (Devlin et al., 2019; Rogers et al., 2020). Unlike prior methods that relied on fixed embeddings or limited context windows, BERT’s bidirectional context allows it to understand a word’s meaning based on the words that come both before and after it. This capacity to model nuanced linguistic features makes Transformer models powerful tools in distinguishing AI-generated content, as they can detect inconsistencies, unnatural phrasing, or contextually implausible terms that simpler models might overlook.

However, while BERT and similar models demonstrate improved performance, they also present challenges and limitations. First, these models are computationally intensive, requiring substantial processing power and memory, which can limit their scalability in real-world detection scenarios (Strubell et al., 2019). Second, BERT’s effectiveness can decline when faced with adversarial paraphrasing, where small, context-preserving modifications to AI-generated text can elude detection (Mosca & Verspoor, 2021). Third, training or fine-tuning Transformer models requires large labelled datasets—something not always available for detecting AI-generated content where ground truth is scarce. Empirical evaluations have shown that while BERT-based classifiers may achieve high accuracy on test sets (e.g., above 90% in controlled conditions), their performance may degrade when applied to texts from newer or more sophisticated AI models (Kreutzer et al., 2022; Uchendu et al., 2020).

#### 2.3.2.2 Contextual Embeddings and Semantic Understanding

Beyond examining individual words, contextual embeddings enable models to represent words within their specific linguistic environment. Models like DeBERTa (Decoding-enhanced BERT with Disentangled Attention) introduce refined attention mechanisms and positional encodings, improving upon BERT’s contextual understanding and offering more robust performance on downstream tasks (He et al., 2020). By capturing deeper semantic relationships, these embeddings help detection algorithms identify subtle anomalies in meaning, coherence, and stylistic consistency across larger spans of text.

Nevertheless, as with BERT, improved embedding techniques do not guarantee foolproof detection. Contextual embeddings can still be misled by clever adversarial strategies, domain shifts, or text that falls outside the distribution of the data on which they were trained. Additionally, developing and fine-tuning these models is resource-intensive and requires careful parameter selection, hyperparameter tuning, and ongoing maintenance.

## 2.4 Text Preprocessing and Feature Engineering

Text preprocessing and feature engineering are critical components of Natural Language Processing (NLP) workflows, particularly in the context of detecting AI-generated content. These processes convert raw text into a more manageable form and extract meaningful characteristics, enabling machine learning models to more accurately identify linguistic markers that distinguish human-written text from artificially generated prose. By standardizing text and highlighting key features, preprocessing and feature engineering enhance model robustness and improve detection performance.

### 2.4.1 Foundational Preprocessing Techniques

#### 2.4.1.1 Tokenization and Normalization

Tokenization involves segmenting text into smaller units—often words or subwords—so that algorithms can examine language patterns at a fine-grained level (Bird et al., 2009; Manning et al., 2008). Through tokenization, detection models gain the ability to identify repetitive usage patterns or unusual token sequences frequently seen in AI-generated text. Normalization follows by converting text into a consistent format, such as lowercasing and removing punctuation, ensuring that variations in capitalization or orthography do not confound analysis (Jurafsky & Martin, 2022; Hoffmann et al., 2022). Together, tokenization and normalization help neutralize superficial cues that may otherwise allow AI-generated content to evade detection.

#### 2.4.1.2 Stopword Removal

Stopwords—common words like “and,” “is,” or “the”—rarely contribute to the semantic content of a text (Bird et al., 2009; Wang et al., 2024). Removing these frequent yet semantically minimal terms helps models concentrate on more informative linguistic features that may indicate AI authorship. By filtering out stopwords, detection algorithms focus attention on the stylistic and lexical markers that are more characteristic of AI-generated prose, improving their discrimination between human and machine-generated text.

#### 2.4.1.3 Stemming and Lemmatization

Stemming and lemmatization are techniques used to reduce words to their base or dictionary forms, ensuring that morphological variants of the same term are treated uniformly (Porter, 1980; Jurafsky & Martin, 2022). For example, “running,” “ran,” and “runs” can be reduced to a single root form, allowing models to recognize them as the same concept. Such uniformity assists in detecting patterns within AI-generated text, including repeated use of certain root words or limited synonym variation, thus enhancing a model’s ability to identify subtle cues of artificial generation.

### 2.4.2 Feature Engineering for AI Detection

#### 2.4.2.1 TF-IDF and Lexical Features

Feature extraction techniques like Term Frequency-Inverse Document Frequency (TF-IDF) quantify the importance of words within a given document relative to a larger corpus (Rajaraman & Ullman, 2011). This method can highlight terms disproportionately overused by AI models. Alongside TF-IDF, lexical features—such as word frequency distributions, vocabulary richness, and positional patterns—help reveal stylistic anomalies (Gehrmann et al., 2019; Opara, 2024). Machine-generated prose may exhibit less lexical diversity and more uniform usage patterns than human writing, enabling detection systems to leverage these differences as reliable indicators of AI generation.

By applying these preprocessing steps and feature engineering techniques, detection systems transform raw input into a structured, information-rich representation better suited for machine learning classification. Although preprocessing and feature extraction alone cannot conclusively determine the origin of a text, they lay the essential groundwork for more advanced analyses. Properly standardized data and informative features enable downstream models—be they statistical classifiers or transformer-based architectures—to make more accurate predictions about whether a given text originates from a human author or an AI system.

## 2.5 NLP Models for AI-Generated Text Detection

The ability to distinguish AI-generated text from human-authored content has become increasingly vital as artificial intelligence systems produce more sophisticated and human-like content. This section explores various Natural Language Processing (NLP) models employed in identifying machine-generated text, emphasizing the distinctions between supervised and unsupervised learning approaches. By focusing on specific model architectures such as Support Vector Machines (SVM), Random Forests (RF), Gradient Boosting methods, clustering algorithms, and transformer-based models, this section delineates the strengths and limitations inherent to each approach.

### 2.5.1 Supervised Learning Models

Supervised learning models are foundational in AI-generated text detection, relying on labelled datasets where each text sample is annotated as either human-written or machine-generated. These models learn to identify patterns and features that distinguish the two classes, enabling accurate classification of new, unseen texts.

#### 2.5.1.1 Support Vector Machines (SVMs)

Support Vector Machines (SVMs) are a well-established choice for text classification tasks due to their ability to handle high-dimensional feature spaces and to find optimal decision boundaries between classes (Manning et al., 2008; Jurafsky & Martin, 2022). In the context of detecting AI-generated text, SVMs often rely on lexical, syntactic, or embedding-based features (Ippolito et al., 2020; Wang et al., 2024). Their capacity to handle a large number of features is beneficial, especially when detecting subtle differences between human-written and machine-generated texts.

However, as AI models become more advanced and produce content that closely mimics human styles, SVMs may struggle to maintain stable decision boundaries. The fixed nature of SVM decision boundaries can lead to decreased performance when faced with evolving text generation techniques that introduce nuanced stylistic variations (Fröhling & Zubiaga, 2021).

#### 2.5.1.2 Random Forests (RF)

Random Forest classifiers, which build ensembles of decision trees, have gained attention in this domain for their robustness and improved generalization (Breiman, 2001; Kreutzer et al., 2022). By randomly sampling subsets of training data and features for each tree, RF models reduce the risk of overfitting—a pervasive challenge in high-dimensional text classification tasks—by ensuring that the final prediction is less sensitive to individual tree biases.

In the context of AI text detection, RF models have demonstrated high accuracy and robustness by leveraging a diverse set of lexical and syntactic features (Fröhling & Zubiaga, 2021; Salman & Al-Jawher, 2024). Studies have shown that RF classifiers often outperform SVMs, particularly as the complexity of AI-generated content increases. For instance, Salman and Al-Jawher (2024) compared SVM and RF models for sentiment analysis and reported that the Random Forest classifier achieved an F1-score of 0.915 compared to 0.845 for the SVM. Similarly, Fröhling and Zubiaga (2021) utilized RF models to detect texts generated by GPT-2, GPT-3, and Grover, finding that RFs provided more stable and higher-performing detections than SVMs, especially as the generation models became more sophisticated.

Moreover, Kreutzer et al. (2022) highlighted that RF classifiers maintain higher accuracy levels under distributional shifts, a critical advantage as AI text generation techniques continuously evolve. These findings collectively suggest that RFs offer greater stability and adaptability compared to SVMs, making them a more reliable choice for ongoing AI-generated text detection efforts.

#### 2.5.1.3 Gradient Boosting Models (e.g., XGBoost)

Gradient Boosting models, such as XGBoost, represent an advanced supervised learning approach that builds predictive models in a sequential manner, where each new model corrects the errors of the preceding ones (Chen & Guestrin, 2016). This iterative refinement allows Gradient Boosting to capture complex patterns and subtle stylistic cues in AI-generated text that simpler methods might miss.

Chandana et al. (2024) demonstrated the efficacy of XGBoost in detecting AI-generated research abstracts, achieving a 10% improvement in detection accuracy over baseline Random Forest models. This enhancement is largely due to XGBoost’s ability to focus on hard-to-classify instances and to model interactions between features more effectively. As AI-generated texts become more sophisticated, Gradient Boosting models like XGBoost provide a robust framework for maintaining high detection accuracy by continuously adapting to new generation patterns.

### 2.5.2 Unsupervised Learning Models

Unsupervised learning models do not require labelled data; instead, inherent patterns and groupings within the data are identified. These models are particularly useful when labelled datasets are scarce or when new, unseen types of AI-generated texts emerge.

#### 2.5.2.1 Clustering Models

Clustering algorithms, such as K-Means and Hierarchical Clustering, group text samples based on similarity metrics derived from linguistic features like word usage, sentence length, and syntactic structures (Bakhtin et al., 2019). In AI-generated text detection, clustering can help identify groups of texts that deviate from typical human writing patterns, flagging them for further analysis.

While clustering does not provide explicit classifications, it serves as a preliminary step in identifying anomalous texts that may warrant closer inspection. Fröhling and Zubiaga (2021) explored the use of clustering in detecting AI-generated content, finding that texts generated by advanced models tend to form distinct clusters separate from human-written texts, thus facilitating effective anomaly detection.

#### 2.5.2.2 Autoencoders

Autoencoders are neural network models that learn to compress and reconstruct data, capturing essential patterns in the process. When applied to text data, autoencoders can identify AI-generated content by measuring reconstruction errors. AI-generated texts often exhibit higher reconstruction errors due to their repetitive or unnatural structures, which differ from human-written texts (Bakhtin et al., 2019; Montero et al., 2021).

By training autoencoders on a corpus of human-written texts, the model learns to reconstruct natural language patterns accurately. Texts that result in significant reconstruction errors are flagged as potential AI-generated content, providing a valuable tool for unsupervised anomaly detection.

### 2.5.3 Semi-Supervised and Hybrid Approaches

Semi-supervised and hybrid approaches combine elements of both supervised and unsupervised learning, leveraging the strengths of each to enhance AI-generated text detection capabilities.

#### 2.5.3.1 Positive-Unlabeled (PU) Learning

Positive-Unlabeled (PU) Learning is a semi-supervised technique that utilizes a small set of labeled data alongside a larger pool of unlabeled data (Salman & Al-Jawher, 2024). In AI-generated text detection, PU Learning can start with a few known examples of human-written and AI-generated texts and then iteratively identify and incorporate additional AI-generated samples from the unlabeled data.

This method is particularly advantageous when labelled datasets are limited, as it reduces the reliance on extensive manual annotation. By gradually expanding the labelled set with high-confidence AI-generated texts, PU Learning improves its detection boundaries, enhancing overall performance and adaptability to new AI text generation techniques.

#### 2.5.3.2 Hybrid Models

Hybrid models integrate multiple learning types to capitalize on their complementary strengths. For instance, a hybrid system might combine a supervised Random Forest classifier with an unsupervised anomaly detection module. The supervised component provides accurate classifications based on labelled data, while the unsupervised component identifies outliers and novel patterns in the data, ensuring that the system remains effective even as AI-generated texts evolve.

Such hybrid frameworks offer robust and adaptable solutions, enabling detection systems to maintain high accuracy and resilience in dynamic environments where AI text generation technologies are continuously advancing.

### 2.5.4 Transformer-Based Architectures

Transformer-based models, such as BERT and its variants, have revolutionized NLP by providing a deep contextual understanding of language (Devlin et al., 2019; Jurafsky & Martin, 2022). These models can be leveraged in both supervised and unsupervised settings for AI-generated text detection.

#### 2.5.4.1 Supervised Transformer Models

When fine-tuned on labelled datasets, transformer models like BERT achieve high classification accuracy due to their ability to capture intricate linguistic nuances and contextual dependencies (Wang et al., 2024). These models excel in distinguishing between human-written and AI-generated texts by leveraging their deep understanding of syntax and semantics.

#### 2.5.4.2 Unsupervised Transformer Models

In unsupervised settings, transformers can be adapted to detect anomalies or deviations from learned text patterns, which may indicate AI generation. For example, models can be fine-tuned to recognize contextual inconsistencies or unnatural phrasing typical of machine-generated content (Zhang et al., 2024). Despite their computational demands, transformer-based models offer unparalleled contextual insight, making them powerful tools in the AI text detection arsenal.

### 2.5.5 Model Comparisons

Comparative studies highlight the relative strengths and weaknesses of different NLP models in AI-generated text detection. Specifically, Support Vector Machines (SVM) and Random Forest (RF) classifiers have been extensively compared in empirical research.

#### 2.5.5.1 Performance in AI-Generated Text Detection

Salman and Al-Jawher (2024) conducted a study comparing SVM and RF models for sentiment analysis, reporting that the Random Forest classifier achieved an F1-score of 0.915 compared to 0.845 for the SVM. This substantial improvement underscores the robustness of RF models in handling complex and high-dimensional feature spaces often encountered in text classification tasks.

Fröhling and Zubiaga (2021) evaluated SVM and RF models in detecting texts generated by GPT-2, GPT-3, and Grover. Their findings indicated that RF classifiers provided more stable and higher-performing detections than SVMs, particularly as the sophistication of AI-generated content increased. This suggests that RFs are better suited to adapt to the nuanced stylistic variations introduced by advanced text generation models.

#### 2.5.5.2 Robustness to Distributional Shifts

Kreutzer et al. (2022) further demonstrated that Random Forests maintain higher accuracy levels under distributional shifts, a critical advantage as AI text generation techniques continuously evolve. In contrast, SVMs showed a more significant decline in performance when exposed to new styles of AI-generated text, highlighting RF’s superior adaptability in dynamic environments.

#### 2.5.5.3 Computational Efficiency and Scalability

While SVMs are effective in high-dimensional spaces and require less computational power compared to ensemble methods, Random Forests offer better scalability and robustness at the cost of increased computational resources. The trade-off between computational efficiency and detection accuracy must be considered when selecting the appropriate model for specific detection tasks.

#### 2.5.6 Summary and Outlook

In conclusion, the detection of AI-generated text benefits from a diverse array of NLP models, each contributing unique strengths to the task. Support Vector Machines (SVMs) and Random Forests (RF) both serve as strong supervised baselines, with empirical evidence suggesting that RFs often maintain higher accuracy and robustness as AI-generated texts grow more challenging to differentiate from human writing. Gradient Boosting methods such as XGBoost can enhance detection accuracy by capturing complex patterns, while unsupervised strategies like clustering and autoencoders are essential for anomaly detection in the absence of labelled data. Semi-supervised and hybrid methods bridge the gap between supervised and unsupervised learning, enabling continuous adaptation to new AI generation techniques. Transformer-based architectures further enrich detection capabilities through deep contextual understanding, albeit with higher computational demands.

As AI-generated text continues to advance, the integration and combination of these diverse models will be crucial in maintaining effective detection systems. Future research should focus on developing hybrid frameworks that leverage the strengths of multiple approaches, ensuring that detection methods remain resilient and accurate in the face of increasingly sophisticated AI text generation technologies.

## 2.6 Machine Learning (ML) in AI-Generated Text Detection

This section explores the role of Machine Learning (ML) in detecting AI-generated content, focusing on ML’s capabilities in recognizing patterns, handling both labelled and unlabeled data, and ensuring adaptability as AI models evolve. Unlike Section 2.5, which covers specific NLP models and their architectures, this section emphasizes broader ML strategies and their practical applications in the dynamic context of AI-generated text detection.

### 2.6.1 Pattern Recognition in AI Detection

ML models are highly effective at recognizing complex patterns in data, which is pivotal for distinguishing human-written text from AI-generated content. Machine-generated text may exhibit telltale signs, such as lexical overuse or repetitive structures, which can serve as detection signals. By training on examples of both human and AI-generated text, ML models can learn to spot these linguistic markers and make accurate predictions about new samples (Mitchell et al., 2023).

Zero-shot detection methods like DetectGPT (Mitchell et al., 2023) have demonstrated that analyzing the probability curvature of a language model can flag AI-generated outputs even without explicitly labelled examples. This approach is particularly advantageous in rapidly evolving scenarios where AI models frequently shift their generative strategies.

### 2.6.2 Handling Labeled and Unlabeled Data

In real-world scenarios, acquiring large volumes of labelled data can be difficult, especially as new AI models emerge. ML addresses this challenge through methods that leverage both labelled and unlabeled data.

#### 2.6.2.1 Semi-Supervised Learning

Approaches such as Multiscale Positive-Unlabeled (MPU) frameworks (Tian et al., 2023) utilize small labelled datasets alongside larger pools of unlabeled data. The model incrementally refines its detection capabilities, which is especially beneficial for short-form AI-generated text (e.g., tweets and reviews).

#### 2.6.2.2 Transfer Learning

Pre-trained models can be fine-tuned for AI detection, leveraging prior knowledge of language structure. Domain adaptation strategies like ConDA (Contrastive Domain Adaptation) adapt detection models to new domains or text genres (Bhattacharjee et al., 2023). These methods help maintain robust performance when the style or domain of AI-generated text diverges from what the model originally encountered.

### 2.6.3 Model Adaptability Over Time

Machine learning solutions must evolve to remain effective against increasingly sophisticated AI text-generation methods. Regular retraining or fine-tuning allows detection models to absorb newly generated text styles. Gradient Boosting algorithms, for instance, incrementally optimize detection by focusing on previously misclassified cases (Chandana et al., 2024). This iterative learning is pivotal in the constantly shifting landscape of AI-generated text.

### 2.6.4 Challenges in Machine Learning for AI Detection

While ML models offer significant benefits, they also face notable challenges. These include the risk of overfitting, data scarcity, and domain shifts.

#### 2.6.4.1 Overfitting

Models risk becoming too specialized in their training data, reducing their ability to generalize. This is especially problematic as AI-generated content continually evolves. A model trained on older examples may fail to recognize newly emergent patterns.

#### 2.6.4.2 Cross-Validation

To mitigate overfitting, practitioners often rely on cross-validation. In k-fold cross-validation, the training set is split into *k-folds*. The model iterates *k* times, each time using a different fold for validation and the remaining folds for training. By averaging performance across folds, cross-validation provides a more reliable estimate of a model’s generalizability (Bates, Hastie, & Tibshirani, 2021). This technique ensures minimal bias in performance metrics, a critical advantage in detection tasks where training data may be limited or unrepresentative.

#### 2.6.4.3 Ensemble Methods

Ensemble methods also help mitigate overfitting by combining multiple models into a unified predictor. Techniques like stacking, bagging, and boosting can produce a more generalized outcome, leading to improved accuracy and robustness (Zhang et al., 2024). By capturing different nuances of language patterns, ensembles reduce the likelihood that any single model dominates the final prediction.

### 2.6.5 Use of Ensemble Methods

Ensemble learning has emerged as a potent strategy for handling the diverse forms of AI-generated content.

#### 2.6.5.1 Bagging

Bagging trains multiple “weak learners” on bootstrap samples, reducing variance in predictions.

#### 2.6.5.2 Boosting

Boosting sequentially learns from misclassifications of prior models, refining detection accuracy—iterative approaches such as Gradient Boosting zero in on challenging examples, enhancing overall performance over multiple rounds.

#### 2.6.5.3 Stacking

Stacking aggregates outputs from base learners (transformer models, SVMs, etc.) into a meta-learner, a process that has shown notable success in AI text detection (Nguyen et al., 2023; Abburi et al., 2023). Research shows that even lightweight ensembles—using only two constituent language models—can match state-of-the-art performance (Abburi et al., 2023), illustrating how ensembling is often more robust than individual models alone.

### 2.6.6 Conclusion

Machine learning lies at the core of AI-generated text detection by recognizing unique text patterns, adapting over time, and effectively leveraging both labelled and unlabeled data. Semi-supervised and transfer learning approaches address data scarcity and evolving AI-generation techniques. Cross-validation helps prevent overfitting, and ensemble methods like stacking, bagging, and boosting enhance generalization. By integrating these ML strategies—ranging from zero-shot probability curvature methods (DetectGPT) to contrastive domain adaptation (ConDA)—researchers and developers can cultivate robust detection systems designed to keep pace with rapid advancements in AI language models (Prova, 2024). Through continual refinement and evaluation, ML ensures that the detection of AI-generated text can remain both accurate and adaptive in an ever-changing digital landscape.

## 2.7 Classification Algorithms and Ensemble Methods for AI-Generated Text Detection

This section focuses on classification algorithms and ensemble methods, emphasizing how these approaches contribute to the performance of detection models for AI-generated text. Unlike Section 2.7, which covered machine learning models more broadly, here we dive into the specific advantages of ensemble techniques—such as voting, stacking, and bagging—and how they enhance model robustness and reliability.

### 2.7.1 Classification Algorithms

Classification algorithms form the basis of AI-generated text detection by categorizing a given text as either AI-generated or human-written. These algorithms range from simple models like Logistic Regression to more complex methods like Support Vector Machines (SVM) and Random Forest. The focus here is not on the individual mechanics of these algorithms, which were discussed earlier, but rather on how their combination through ensemble methods leads to enhanced performance.

### 2.7.2 Ensemble Methods

Ensemble learning is a strategy where multiple models are combined to produce a more accurate prediction than any individual model alone. This section explores the ensemble techniques used in AI-generated text detection, including voting, stacking, boosting, and bagging.

#### 2.7.2.1 Voting

Voting is a straightforward ensemble method where predictions from multiple models are combined, and the final prediction is determined based on a majority vote or an average of the outputs. In AI detection, combining classifiers like Random Forest and SVM can yield more reliable results, as each model contributes its strengths, effectively reducing the impact of individual model biases. Voting can be either hard voting (majority vote) or soft voting (average of predicted probabilities), both of which help in producing balanced and robust predictions (Agarwal, 2023; Chandana et al., 2024).

#### 2.7.2.2 Bagging (Bootstrap Aggregating)

Bagging is an ensemble technique that involves training multiple instances of the same model on different subsets of the training data. By doing so, bagging reduces variance and helps prevent overfitting, which is crucial for detecting evolving AI-generated patterns. For instance, using Random Forest—a bagging-based ensemble of decision trees—has proven effective for text classification because each tree is trained on a random sample of the data, leading to diverse perspectives and reducing the chance of overfitting to specific data points (Şevgi̇n, 2023; Alamleh et al., 2023).

#### 2.7.2.3 Boosting (e.g., Gradient Boosting and XGBoost)

Boosting is a method where models are trained sequentially, each new model correcting the errors made by the previous ones. Techniques like Gradient Boosting and XGBoost are highly effective for AI-generated text detection as they build on weak learners to create a stronger overall model. Boosting works by focusing on data points that were previously misclassified, gradually improving performance. This approach is especially useful for detecting subtle differences between human and AI-generated text, as the iterative process sharpens the model’s sensitivity to small anomalies (Chandana et al., 2024).

#### 2.7.2.4 Stacking

Stacking is a more complex ensemble technique where multiple base learners are trained, and their outputs are used as inputs for a meta-learner. The meta-learner, often a simple model like Logistic Regression, learns how to best combine the outputs from the base models to make a final prediction. In AI detection, stacking is particularly beneficial because it allows the integration of diverse learning approaches—such as combining transformers, decision trees, and clustering models—into a single framework (Yeturu et al., 2023; Zhang et al., 2024).

### 2.7.3 Benefits of Ensemble Methods in AI Detection

The primary advantage of using ensemble methods is the increased accuracy and robustness they bring to detection models. Individual models, while capable, often have inherent limitations, such as biases or overfitting tendencies. By combining multiple models, ensembles reduce these weaknesses and produce more generalized, effective detection systems. Techniques like bagging and boosting also provide mechanisms to handle different types of errors, making the resulting models better suited for the complexities of AI-generated text (Chandana et al., 2024; Zhang et al., 2024).

### 2.7.4 Challenges with Ensemble Methods

Despite their advantages, ensemble methods can present notable drawbacks, most prominently in terms of computational complexity and training time. Training multiple models and combining their results requires more computational resources and can lead to slower inference.

For example, *Nguyen et al. (2023)* demonstrate a stacking ensemble of transformer-based models for AI-generated text detection. Although this approach significantly boosts accuracy over single models, it nearly doubles the overall training time due to each transformer demanding substantial GPU memory and processing power. Similarly, *Chang et al. (2023)* investigate ensemble strategies in real-time AI detection scenarios, observing memory overhead and latency issues that can turn into bottlenecks in large-scale or streaming applications.

Furthermore, ensembles that incorporate resource-intensive models (e.g., large transformers) multiply the computational requirements, making the trade-off between performance gains and computational costs vital. *Kim and Park (2023)* highlight that while ensemble methods can substantially improve detection performance, they often require a careful balance: the increased complexity may not always be justifiable in lower-stakes or resource-constrained environments.

However, the trade-off is often considered worthwhile in high-stakes contexts—such as academic integrity checks or real-time misinformation screening—where accuracy and robustness are paramount. *Alamleh et al. (2023)* argue that despite the added expense, ensemble methods deliver a level of reliability that can be critical in detecting rapidly evolving AI-generated content.

### 2.7.5 Conclusion

Ensemble methods play a crucial role in enhancing the performance of AI-generated text detection models. By combining multiple classifiers through voting, bagging, boosting, and stacking, these approaches provide the robustness and adaptability needed to handle the evolving challenges posed by advanced AI language models. The ability of ensemble techniques to improve generalization and mitigate the limitations of individual models makes them an essential component of the modern AI detection toolkit—particularly in contexts where high accuracy justifies higher computational costs.

## 2.8 Length-Sensitive Detection Techniques

This section explores Length-Sensitive Detection techniques for AI-generated content, focusing on how they manage challenges specific to long-form text. Length-sensitive methods are designed to address the unique characteristics of longer content, such as variability in style, coherence, and the ability of AI to evade detection through sophisticated techniques like paraphrasing or adversarial modifications.

### 2.8.1 Handling Long-Form Content

One of the main challenges with detecting AI-generated text is effectively managing long-form content, such as essays or news articles. These texts have greater variability in tone, style, and complexity compared to short-form content, making them more challenging for traditional detection algorithms. Length-sensitive detection techniques aim to maintain consistency in detection accuracy even when dealing with extended texts. Transformer-based models like Longformer have proven effective at handling these kinds of lengthy texts due to their extended attention mechanisms, allowing the model to capture dependencies across a much larger input sequence than standard transformer models (Li et al., 2024).

### 2.8.2 Adversarial Robustness in Long Texts

Long-form AI content is often susceptible to adversarial attacks, where minor changes are made to evade detection. Adversarial robustness in the context of length-sensitive detection involves ensuring that detection models can still accurately classify AI-generated content, even if adversarial paraphrasing or subtle stylistic changes are applied. Techniques like gradient masking and adversarial training are used to make models more resilient. For instance, J-Guard, a journalism-guided adversarial detection framework, integrates stylistic cues from journalism to strengthen detection, particularly for long-form news content, making it less vulnerable to subtle changes aimed at bypassing detection (Kumarage et al., 2023).

### 2.8.3 Dealing with Paraphrasing in Long-Form Text

Paraphrasing is a common evasion tactic, especially in longer AI-generated texts, where the content can be subtly altered to mimic human creativity. Length-sensitive detection methods need to account for this by not just relying on simple word or phrase matches but by understanding the overall semantic coherence and writing style. Approaches like contextual embeddings from transformer models help detect paraphrased content by comparing the semantic meaning of the original and altered texts. Additionally, DeTeCtive, a framework utilizing multi-level contrastive learning, has shown effectiveness in detecting AI-generated content by focusing on distinguishing writing styles, which is crucial when dealing with paraphrased or slightly modified long texts (Guo et al., 2024).

### 2.8.4 Hybrid Models for Long Content

Hybrid models are particularly useful for handling the complexity of long-form AI-generated content. By combining machine learning techniques with rule-based approaches, hybrid models can leverage the best of both worlds: the flexibility of machine learning in recognizing complex, non-linear relationships and the interpretability of rule-based systems. For example, using TF-IDF for initial feature extraction combined with a deep learning model for contextual understanding can improve the detection of AI-generated long texts. This combination helps maintain consistency in identifying unique markers that differentiate machine-generated text from human writing across longer sequences (Zhang et al., 2024).

### 2.8.5 Scalability Challenges

Length-sensitive detection also involves addressing the scalability challenges associated with processing long texts. Models like Longformer and BigBird were developed to handle the large computational requirements associated with long-form content by using sparse attention mechanisms, which reduce the complexity compared to the full attention mechanism of standard transformers. These models can efficiently process longer texts, making them ideal for real-world applications where large volumes of data must be analyzed for AI-generated content (Kramp et al., 2023; Li et al., 2024).

### 2.8.6 Application in Journalism and Academia

Length-sensitive detection techniques are particularly crucial in fields like journalism and academia, where the integrity of long-form content is paramount. AI-generated long-form articles pose a significant risk of misinformation and academic dishonesty. By employing specialized models that focus on maintaining coherence, stylistic consistency, and robustness against evasion techniques, these sectors can better safeguard their content from AI-related manipulations (Zagorulko, 2023; Eke, 2023).

### 2.8.7 Conclusion

Length-sensitive detection techniques thus play a critical role in maintaining the accuracy and reliability of AI detection systems when dealing with extended content. By leveraging advanced transformer models, ensuring adversarial robustness, and utilizing hybrid approaches, these methods help address the unique challenges posed by long-form AI-generated texts, thereby supporting the integrity of information in fields where content length and complexity are significant factors.

## 2.9 Academic Integrity and Ethical Concerns

This section addresses the ethical implications and academic integrity issues arising from the use of AI-generated content, particularly within educational settings. As AI-generated text becomes increasingly sophisticated, it is essential to discuss how detection tools can be leveraged to maintain academic standards and ensure ethical practices. Unlike previous sections that focused on detection methodologies, here we emphasize the societal impacts, educational policies, and necessary guidelines to handle AI-generated content responsibly.

### 2.9.1 Impact on Academic Integrity

The use of AI tools like ChatGPT has raised significant concerns in academic environments, especially in maintaining the originality and authenticity of students’ work. Students can use AI to generate essays and assignments or even answer exam questions, which threatens to undermine the educational process by allowing them to submit work that is not genuinely theirs. The widespread use of AI-generated text poses a risk of encouraging plagiarism and diminishing the value of learning, as it becomes harder to assess students’ true understanding and capabilities.

Recent surveys underscore these concerns; for instance, Jiao et al. (2024) highlight how AI-assisted cheating is increasingly prevalent in computing education. They report that 31% of surveyed instructors had encountered suspiciously uniform writing structures consistent with AI outputs. Additionally, UNESCO (2023) emphasizes the importance of creating institutional frameworks to preserve academic honesty in the face of rapidly advancing generative AI technologies.

### 2.9.2 Role of Detection Tools

Effective detection tools play a vital role in preserving academic integrity. Tools like OpenAI’s AI Text Classifier and DetectGPT provide practical means for educators to verify whether submitted assignments have been generated by AI. These tools are being increasingly integrated into institutional workflows to catch instances of AI-assisted plagiarism and deter misuse. However, it is also essential to understand the limitations of these tools, as they are not infallible and can sometimes produce false positives or fail to recognize sophisticated AI manipulations.

Wang et al. (2023) highlight a growing trend in universities where reliance on commercial AI-detection services is tempered by concerns over false positives and lack of transparency in detection algorithms. The authors note that while detection tools serve as a first line of defence, manual review and educator training remain crucial for maintaining fairness and accuracy.

### 2.9.3 Educational Policies and Guidelines

Academic institutions must adapt to the rising use of AI by creating clear guidelines that define the acceptable use of AI tools in coursework and research. Establishing policies that specifically address AI-generated content is crucial for setting expectations for both students and educators. For instance, some universities have begun requiring students to declare any AI assistance in their work. This approach promotes transparency and encourages the responsible use of AI technologies.

Jiao et al. (2024) provide a global overview of institutional responses to generative AI. Their study reveals that over 60% of surveyed universities have introduced or updated academic honesty policies regarding LLM usage, signalling a collective push toward codified guidelines that clarify AI’s permissible role in student work.

### 2.9.4 Training and Awareness

Beyond detection and policy, education plays a key role in ensuring academic integrity in the era of AI. Educators need to be trained to use AI detection tools effectively and to understand the ethical implications of AI in student work. Students also need to be educated on the ethical boundaries of using AI, emphasizing that while these tools can aid in learning, they should not replace the process of original thought and personal effort. Encouraging students to view AI as a supplement rather than a substitute for learning can help mitigate ethical concerns.

A novel approach was proposed by Vasilatos et al. (2023), who developed the HowkGPT system for context-aware perplexity analysis of student homework. Their pilot study shows that 73% of educators who received specialized training on AI and plagiarism detection felt more confident distinguishing AI-generated material from genuine student submissions, illustrating the importance of educator preparedness.

### 2.9.5 Societal Implications and Journalistic Perspective

The ethical concerns around AI-generated content extend beyond academia into broader societal contexts. Misinformation, fake news, and deepfake content can all be fueled by sophisticated generative AI models. In the journalistic realm, the core issue lies in preserving public trust in media outlets.

Reuters Institute (2024) indicates that AI-generated news pieces have appeared in local and online news outlets without clear disclosure, raising significant credibility challenges. Many journalists worry that AI’s potential for rapid content generation could amplify misinformation when not properly vetted or labelled (Wu et al., 2024). The Tow Center for Digital Journalism (n.d.) has similarly reported that AI-authored news articles, while occasionally flagged by editorial teams, often slip through when production demands are high, risking the publication of unverifiable or manipulated stories.

Prakash Jambunathan et al. (2024) highlight how detection frameworks initially designed for academic contexts can be adapted for real-time news verification. Their ConvNLP approach extends beyond classroom applications, identifying AI-generated paragraphs in “breaking news” scenarios with 89% accuracy. By filtering out machine-authored or manipulated text, journalists can maintain credibility and foster public trust.

### 2.9.6 Balancing Innovation and Integrity

While there are genuine ethical concerns regarding AI-generated content, it is also important to acknowledge the potential benefits of these tools. AI can assist students in brainstorming ideas, providing language support, or helping them understand complex topics. Similarly, in journalism, automated content generation can free up reporters to focus on more investigative work or personalize stories for local audiences.

The key is balancing the use of AI to enhance output without compromising integrity. Academic and media policies must evolve to support responsible innovation while retaining a firm stance against misuse. UNESCO (2023) advocates for frameworks allowing the controlled adoption of AI technologies, ensuring that human oversight and editorial standards remain the final authority in both educational and journalistic processes.

### 2.9.7 Conclusion

Academic integrity and ethical concerns are central to discussions on AI-generated content in both educational and journalistic settings. By leveraging detection tools, implementing transparent guidelines, and promoting awareness about the responsible use of AI, institutions can navigate the challenges posed by these technologies. Ensuring that AI serves as an aid rather than a shortcut is paramount to preserving the value and credibility of academic achievements and safeguarding public trust in media.

### 2.10 Filtering AI-Generated Content

Filtering AI-generated content is essential for preserving training data quality and ensuring reliable AI model performance. Recent research shows that recursive training on AI-generated text can lead to model collapse, where performance degrades over successive iterations (Shumailov et al., 2024). This section explores why filtering matters, how to implement it, and under what conditions it is most beneficial.

### 2.10.1 Importance and Risks of Recursive Training

AI-generated text is often repetitive or follows predictable patterns. When such data seeps into training sets, it can bias future models and create feedback loops in which models learn from their own outputs (Chen et al., 2024; Briesch et al., 2023). Studies have found that excessively synthetic training data leads to nonsensical outputs (Wenger, 2024). These phenomena—sometimes called “self-consuming training loops” or “the curse of recursion”—underscore the need to filter out AI-generated content before it contaminates training sets (Shumailov et al., 2023).

### 2.10.2 Methods for Filtering AI-Generated Content

#### 2.11.2.1 Metadata Analysis

Examining creation timestamps, software versions, and other markers can help flag suspicious entries. Although not foolproof, it serves as a first line of defence (Uzun, 2023).

#### 2.11.2.2 Detection Models

Classifiers like DetectGPT or OpenAI’s AI Text Classifier can filter AI-generated text by spotting telltale patterns (Mitchell et al., 2023). These models reduce the risk of inadvertently adding synthetic content to training sets.

#### 2.11.2.3 Human-in-the-Loop

The human review remains indispensable for verifying flagged content. Even advanced classifiers can fail against newer or fundamentally different AI architectures (Bakhtin et al., 2019).

### 2.11.3 Mitigating Model Collapse

Excessive reliance on synthetic data can lead to model forgetfulness and degrade generative diversity (Shumailov et al., 2023). Hybrid approaches—accumulating real and synthetic data—may offset some risks (Gerstgrasser et al., 2024). Regular dataset audits and multiple independent detection pipelines further ensure that feedback loops do not spiral into model collapse (Stray, 2023).

### 2.11.4 Ethical and Practical Considerations

#### 2.11.4.1 Ethical Filtering

Deciding whether and how AI-generated content belongs in certain datasets depends on context (e.g., journalism, academia). Striking a balance between leveraging AI outputs and retaining dataset authenticity is critical (Eke, 2023).

#### 2.11.4.2 Transparent, Provable Evidence

Effective filtering must substantiate why the content was flagged as AI-generated. Clear audit trails and transparent detection results build stakeholder confidence and reduce costly errors.

### 2.11.5 Conclusion

Filtering AI-generated content prevents performance degradation and safeguards the integrity of training datasets. By blending metadata analysis, detection models, human oversight, and regular audits, organizations can better avoid the pitfalls of recursive training and model collapse (Shumailov et al., 2024; Briesch et al., 2023; Wenger, 2024). This proactive strategy not only improves model quality but also ensures stakeholder confidence in the reliability and provability of detection outcomes.

# 3. Design Chapter

## 3.1 Introduction

This chapter presents the design for a proof-of-concept system for detecting AI-generated text. The core objective is to develop a system capable of transforming human-written articles into paraphrased AI-generated versions of them and effectively distinguishing between the two of them. This design primarily leverages an existing dataset, specifically the “All the News 2.0” dataset (Components One, 2022), to simulate realistic detection scenarios. Unlike many similar systems that rely purely on classification, this project aims to investigate the intricacies of content manipulation using advanced Generative AI models and explore robust feature extraction, detection methodologies, and explainability aspects that make model decisions transparent to stakeholders.

## 3.2 Data Acquisition and Preprocessing

### 3.2.1 Dataset Choice and Citation

The “All the News 2.0” dataset, published by Components One (2022), featuring over 2.7 million news articles from diverse sources, including mainstream and niche publications. This breadth helps ensure exposure to varied writing styles—critical for robust AI detection.

Dataset Volume and Selection

* Total Articles: ~2.7 million
* Planned Usage: Approximately 50,000 articles for practical constraints (due to hardware and processing time).
* Selection Strategy:
  + Random Sampling to cover a wide range of publication dates and categories (politics, business, science, etc.).
  + Topic Stratification: Aim for balanced representation across major topic areas (e.g., ~10k per category) for variety.

By using only a subset (~2%) of the full dataset, plans to manage computational resources while retaining sufficient diversity to stress-test detection models.

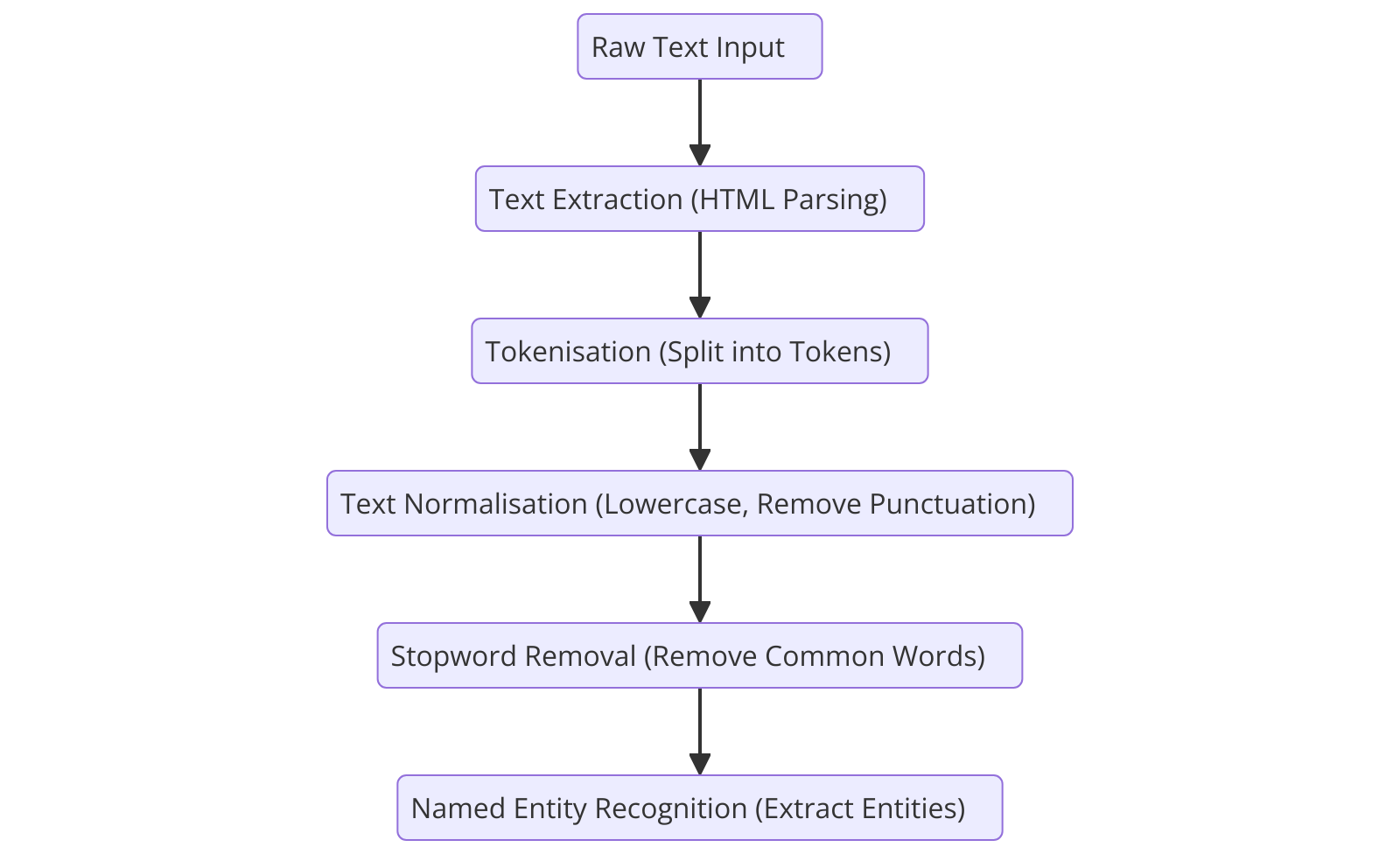
**3.2.2 Preprocessing Workflow**

The preprocessing pipeline ensures data quality for both the generative (AI paraphrasing) and detection (model training) phases.

1. Text Extraction
   * Strip away non-essential metadata (e.g., author names, publication date) to focus on the article body.
   * *Note*: Named Entity Recognition (NER) (Lin et al., 2024) might help remove personal details or locations from texts to prevent misleading context.
2. Normalization
   * Lowercasing, stemming, or lemmatization (Paaß & Giesselbach, 2023).
   * Example: “Running,” “ran,” and “runs” → root form “run.”
3. Stopword Removal
   * Eliminates frequent but semantically minimal words (“and,” “the,” “is”), focusing on content-bearing tokens.
   * SpaCy offers optimized pipelines (multi-threading) for large data (Petrov et al., 2023).

Hardware Considerations:

* Due to budget constraints, plan to run preprocessing on a laptop with an NVIDIA RTX 4060 GPU (Alienware M18). This mid-range setup can handle ~50k articles but demands careful batching and memory management.



The preprocessing stage includes a set of distinct operations aimed at optimising data quality for AI paraphrasing and subsequent detection (Camacho-Collados & Pilehvar, 2017). Text extraction involves retrieving only the main body of articles to eliminate noise that may negatively impact both the generation and detection processes. In a practical scenario, a nNamed Eentity Rrecognition (NER) model could be implemented to detect and filter out non-content information like author names, publication dates, or locations mentioned in metadata (Lin et al., 2024). This ensures that only relevant content contributes to training and testing the detection model, enhancing its effectiveness by focusing purely on the stylistic and semantic features of the text.

Text normalisation techniques, such as lowercasing and stemming or lemmatisation, are applied to reduce linguistic variability without sacrificing meaning (Paaß & Giesselbach, 2023). For instance, “running” and “ran” are lemmatised to a common root form, enabling the model to better capture underlying themes and reduce noise from superficial lexical variations. This step is critical in reducing the size of the feature space, making models more computationally efficient while maintaining semantic consistency.

Removing stopwords like “is,” “the,” and “and” optimises the input for both AI generation and detection models by emphasising content-bearing words rather than grammatical constructs (Kaushal & Mahowald, 2022). The stopword removal process will be executed using established NLP libraries, with SpaCy offering more flexibility for handling large datasets efficiently through its optimised pipelines (Petrov et al., 2023). This choice ensures that the system can process large volumes of data quickly, which is crucial for scalability.

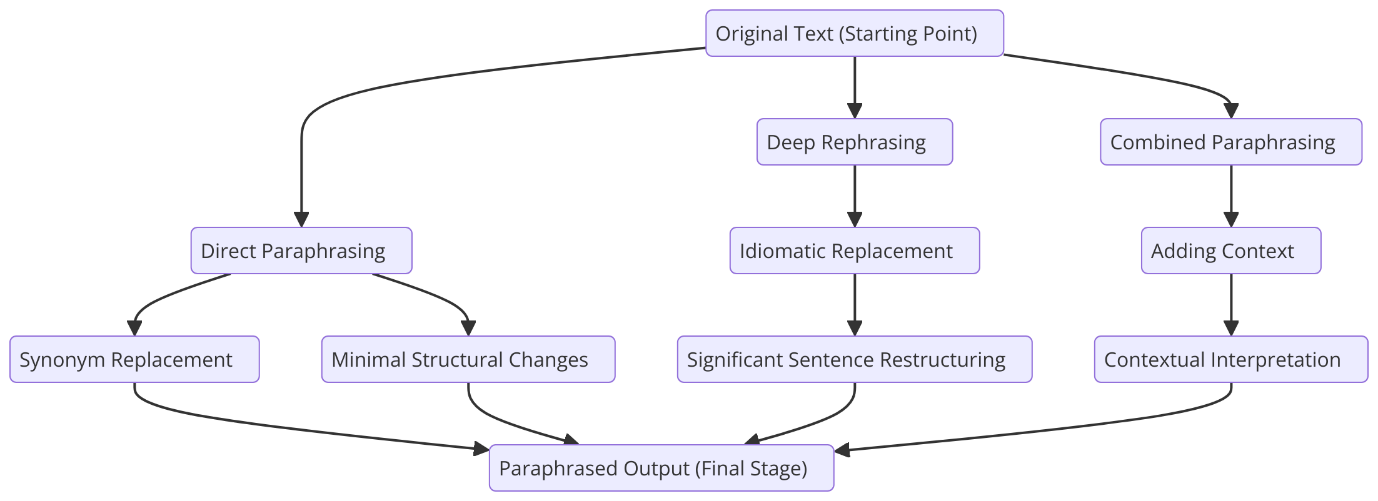
**3.3 Dataset Paraphrasing using AI**

The Generative AI model employed for content paraphrasing is GPT-4 due to its advanced language comprehension capabilities, which allow for nuanced manipulation of the text while preserving its core message (OpenAI, 2023). The decision to use GPT-4, specifically via OpenAI’s API, is rooted in its high-quality text generation, ensuring that the paraphrased output provides a genuine challenge to detection algorithms (OpenAI, 2023).

The paraphrasing process will be multi-layered to simulate different levels of content transformation (Li et al., 2019). Direct paraphrasing involves minimal transformation, where GPT-4 rephrases sentences using synonyms or slight structural modifications (Zhou & Bhat, 2021). This type of paraphrasing represents a basic but realistic form of content manipulation, reflecting scenarios where minimal effort is made to evade detection.

Deep rephrasing, on the other hand, involves significant alterations, including changing sentence structure, modifying idiomatic expressions, and introducing alternative phrasings that add depth to the reworded content (Wahle et al., 2023). Such paraphrasing challenges the detection model’s ability to understand semantics rather than merely identifying lexical changes.

Combined paraphrasing and augmentation involves adding additional context or reinterpreting information slightly, resulting in enhanced content that not only paraphrases but augments the original (Li et al., 2019). This type represents an even more sophisticated form of manipulation, akin to what may be seen when an AI attempts to obscure its origin by adding noise or additional, loosely related context.



One concern with multi-iteration paraphrasing is the risk of creating an imbalanced dataset. Repeated paraphrasing iterations could introduce biases, where AI-generated content becomes overrepresented in comparison to the original human-written content (Wahle et al., 2023). To address this, a balanced approach will be maintained by ensuring that the ratio of human to AI-generated articles remains consistent (Zhou & Bhat, 2021). Additionally, careful sampling techniques will be employed to mitigate any skewness in the dataset, allowing the models to generalise effectively without being disproportionately exposed to paraphrased content.

**3.4 System Architecture**

The system architecture involves three core modules: Data Preprocessing, AI Content Generation, and Detection Module. Each module is discussed in detail below, covering its purpose, the tools used, and how it integrates into the larger detection pipeline.

The Data Preprocessing Module encompasses text extraction, normalisation, and stopword removal, implemented primarily using Python libraries like Pandas, NLTK, and SpaCy (Bird et al., 2009). The use of SpaCy over NLTK for preprocessing larger datasets is justified by its optimised tokenisation capabilities and multi-threaded support, which ensures efficient and faster processing, especially important for a dataset comprising millions of articles (Choi et al., 2015).

For the paraphrasing of original articles, GPT-4 is employed via OpenAI’s API. The reason for choosing GPT-4 is its superior natural language capabilities that help create paraphrased content that is not only human-like but also lexically and syntactically diverse (OpenAI, 2023). This module utilises multiple configurations of temperature and repetition penalties. Varying the temperature affects the diversity of the text (Holtzman et al., 2020). A low-temperature setting (e.g., 0.2) will produce more deterministic output, which may closely resemble the original text in terms of coherence but with a modified form (Holtzman et al., 2020). Conversely, a higher temperature (e.g., 0.8) introduces randomness, creating more diverse paraphrasing, which is essential for stress-testing the detection system against varied AI outputs.

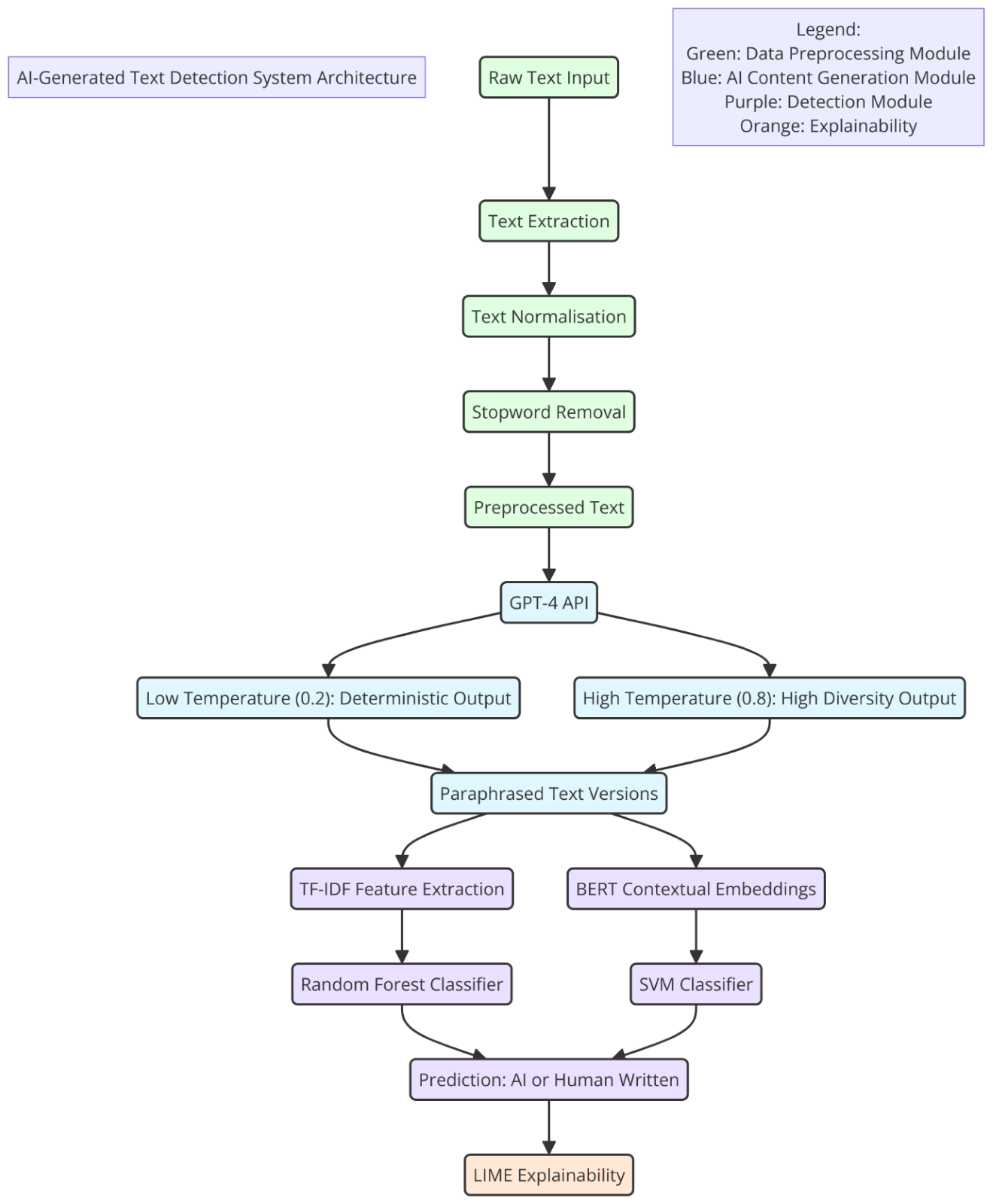
Multi-iteration paraphrasing will involve paraphrasing each text multiple times to create varied versions. However, to prevent imbalance, the system will ensure that these iterations are distributed proportionally. For instance, only a subset of human-written articles will be paraphrased multiple times, while maintaining a representative sample of unaltered articles. By doing so, the model is exposed to intra-sample diversity without overrepresenting the AI-generated content.

The Detection Module integrates hybrid detection techniques, combining traditional NLP-based feature extraction with advanced machine-learning classifiers to identify whether text is AI-generated or human-written (Jawahar et al., 2019). TF-IDF is used to generate lexical features that can highlight repetitive and predictable language usage common in AI-generated text (Alamleh et al., 2023). TF-IDF has been chosen for its simplicity and effectiveness in providing insight into the significance of terms within the context of each document compared to the entire corpus.

Machine learning classifiers like Random Forest (RF) and Support Vector Machine (SVM) are utilised for classification. RF is employed due to its robustness against overfitting, which is particularly relevant when dealing with high-dimensional textual data (Alamleh et al., 2023). It can effectively handle non-linearity and interactions among variables, such as the frequency and co-occurrence of words. SVM is used alongside RF due to its proven effectiveness in high-dimensional spaces, especially for text classification tasks involving sparse matrices resulting from TF-IDF features.

To deepen the semantic understanding of the content, BERT (Bidirectional Encoder Representations from Transformers) is also integrated (Devlin et al., 2019). Unlike TF-IDF and traditional machine learning classifiers, which primarily operate on surface-level linguistic features, BERT captures contextual relationships by considering the bidirectional context of words within sentences. This ensures that more subtle stylistic differences are accounted for, providing a more holistic approach to distinguishing AI-generated content.

A hybrid approach combining Random Forest, SVM, and BERT provides a multi-faceted detection capability—surface-level features are captured through TF-IDF, while deep contextual features are analysed through BERT (Alamleh et al., 2023). Such a combination ensures robustness in diverse scenarios of paraphrasing and effective handling of both linear and non-linear data relationships. Additionally, explainability will be integrated using Local Interpretable Model-agnostic Explanations (LIME) to provide transparency about which features contribute most to each classification decision, ensuring the system is understandable for stakeholders, especially in sensitive areas such as journalism and academia (Ribeiro et al., 2016).



**3.6 Tools and Technologies**

The project employs a combination of tools and technologies, each selected based on specific requirements that cater to the distinct stages of data manipulation, text generation, and detection. Python is chosen for its extensive ecosystem of libraries, popularity in NLP, and versatility in both research and industry. Python’s rich repository of machine learning and NLP libraries enables seamless integration of all required tools and reduces development time, making it the backbone of the system’s implementation (McKinney, 2018).

Scikit-learn will be used for implementing traditional machine learning models, such as Random Forest and SVM, and evaluating them using established metrics like precision, recall, and F1-score (Géron, 2022). TensorFlow or PyTorch will be used for deep learning models like BERT, offering GPU acceleration capabilities that are critical when processing and training models on large-scale text corpora. Additionally, LIME will be integrated to provide interpretability by explaining model decisions, crucial for understanding the rationale (Ribeiro et al., 2016).

The OpenAI API is central to integrating GPT-4-based text paraphrasing. GPT-4 was chosen for its sophisticated understanding of human-like text, enabling nuanced paraphrasing that challenges detection models significantly (OpenAI, 2023). The use of the OpenAI API provides the flexibility to adjust hyperparameters—such as temperature and top-p—that influence the text output, which is vital for simulating various levels of content manipulation. By modulating these parameters, paraphrased content can vary from predictable and close to the original text to highly diverse, making the detection task more challenging.

Pandas and Numpy are pivotal in managing large-scale datasets. Pandas is used for its high-level data manipulation capabilities, including filtering, transformation, and merging of datasets, which is essential when dealing with millions of records (McKinney, 2018). Numpy underpins numerical operations, optimising the speed of matrix calculations that are vital in text processing, ensuring efficient utilisation of computational resources during both the preprocessing and feature extraction stages.

SpaCy is used for NLP preprocessing due to its speed and efficiency in tokenising large text datasets. It outperforms traditional libraries like NLTK when handling big data, providing an optimised pipeline for operations such as tokenisation, lemmatisation, and named entity recognition. Its multi-threading capabilities further enhance performance, particularly important given the project’s large-scale data requirements (Choi et al., 2015).

For the machine learning component, Scikit-learn is the primary tool for implementing traditional classifiers, such as Random Forest (RF) and Support Vector Machine (SVM). Scikit-learn offers robust implementations of these models, with extensive support for hyperparameter tuning and model evaluation, which is crucial for ensuring optimal classifier performance (Géron, 2022). The Random Forest classifier is selected for its ability to handle large feature spaces and mitigate overfitting, making it suitable for the complexity of the dataset. SVM is chosen due to its effectiveness in high-dimensional text classification, providing a complementary perspective to Random Forest by focusing on optimal hyperplane separation for different classes.

For deep learning models, TensorFlow or PyTorch are used to implement BERT. These frameworks are selected for their GPU acceleration capabilities, allowing for efficient training and inference on large text datasets. BERT is integrated to provide a semantic understanding of text, capturing context at a deeper level than traditional NLP techniques (Devlin et al., 2019). TensorFlow and PyTorch support fine-tuning BERT on specific tasks, ensuring that the model adapts to the nuances of the dataset and improves classification accuracy for paraphrased versus human-written text.

The combination of these tools provides a comprehensive technology stack that supports a nuanced approach to paraphrasing and detecting AI-generated content. Each tool is selected for its unique strengths, contributing to different aspects of the overall detection system—from efficient data handling and preprocessing to sophisticated machine learning and deep learning implementations.

## 3.7 Functional & Non-Functional Requirements

To facilitate clear project endpoints and evaluation criteria, will define high-level requirements:

### 3.7.1 Functional Requirements

1. FR1: The system shall parse, preprocess, and store a subset (~50k) of news articles from “All the News 2.0.”
2. FR2: The system shall generate multi-tier paraphrased AI content (direct, deep, combined) using GPT-4.
3. FR3: The detection module shall classify content as AI- or human-written with at least 80% F1-score.
4. FR4: The system shall provide explainability via LIME visualizations for each classification.

### 3.7.2 Non-Functional Requirements

1. NFR1: Performance: The system should preprocess the dataset within 12 hours on an RTX 4060 GPU laptop.
2. NFR2: Scalability: The architecture must accommodate future real-time pipelines (Kafka).
3. NFR3: Maintainability: Code should be modular (Data Preprocessing, Generation, Detection) to allow easy updates.
4. NFR4: Interpretability: The detection module should produce human-readable explanations for at least 95% of flagged texts.

## 3.7 Summary

The design chapter has provided an in-depth explanation of the architecture, methodologies, and tools involved in the creation of an AI-generated text detection system. Each component was carefully selected based on its strengths in addressing the unique challenges posed by paraphrased AI-generated content, ensuring robustness and scalability for future development phases.

The focus has been on creating a proof-of-concept system with an architecture that allows easy expansion into a fully deployed solution next semester. By combining state-of-the-art language models, traditional NLP techniques, machine learning classifiers, and explainability tools, the design aims to ensure a comprehensive approach to tackling the growing challenge of distinguishing between human-written and AI-generated content.

# Chapter 4: Implementation

**4.1 Technical Setup and System Architecture.**

[GitHub repository](https://github.com/MichaelShpyl/AI-Text-Detection-Tool)

The implementation phase translates the design into a working system, encompassing data processing pipelines, model training, and user-facing applications. All components were developed and executed on a modest hardware setup: a single machine with an NVIDIA GPU (with limited VRAM) and standard CPU/RAM resources (no specialized high-end server was available). The software environment was built in Python, leveraging the PyTorch deep learning framework and Hugging Face Transformers library for model development. The overall architecture consists of distinct modules for data acquisition, model training, interpretability, and user interface integration. These modules communicate through well-defined interfaces. For instance, the detection model is exposed via a local API endpoint that the user interface (a browser extension and a web dashboard) can query. This modular architecture ensures that each part of the system (data, model, explanation, UI) can be developed and tested in isolation and later integrated. A high-level schematic of the system architecture is illustrated in Figure 4.1, which shows how data flows from raw text input (either user-provided or from an article) through the trained AI detection model and then to the output components (dashboard visualization or browser overlay). The implementation adheres to this architecture, with careful attention to reproducibility (all experiments were logged and can be reproduced with fixed random seeds and documented configurations) and scalability (the code is structured so that improvements, such as using a more powerful GPU or distributed computing, can be incorporated with minimal refactoring).

**4.2 Dataset Construction Using the DeepSeek-R1 Model**

In this section, the construction of a bespoke dataset for AI-generated text detection is described. All AI-generated content in the dataset was produced exclusively with the DeepSeek-R1 1.5B language model. This approach replaced initial plans to use OpenAI’s GPT-3.5, in order to allow full offline control of generation and avoid API constraints. The choice of DeepSeek-R1 1.5B over larger alternatives is first justified, followed by a description of the prompt design and evaluation procedure, and an explanation of how human news articles were transformed into AI-paraphrased and AI-generated texts. The resulting dataset was carefully curated with balanced classes and an 80/10/10 train–validation–test split.

**4.2.1 Model Selection and Rationale (DeepSeek 1.5B vs 7B)**

The DeepSeek-R1 model is a transformer-based language model available in different parameter scales (1.5 billion vs. 7 billion). Preliminary DeepSeek model testing was conducted to determine which version best suited the dataset generation needs. The 7B variant was expected to potentially produce higher-quality text due to its larger size but at the cost of slower inference. In experiments, the output quality of DeepSeek-R1 7B and 1.5B was found to be comparable in semantic fidelity and coherence, as measured by metrics like BERTScore F1 (which assesses semantic similarity to a reference) and qualitative review of text fluency. Crucially, however, the 1.5B model was roughly 3× faster than the 7B model for paraphrasing tasks. This speed-up was confirmed by timing the generation of identical outputs with each model – the 1.5B model consistently produced results about one-third of the time. Given the need to generate a large volume of text (hundreds of thousands of articles) within a limited time frame, this performance trade-off favoured the smaller model. Consequently, DeepSeek-R1 1.5B was selected as the generation engine for all dataset augmentation, as it offered near-equal output quality with significantly better efficiency. By choosing DeepSeek (an open-source model), full control over the generation pipeline was maintained and the rate limits and costs associated with the GPT-3.5 API were avoided, without sacrificing the realism or correctness of the generated text.

**4.2.2 Prompt Design for Paraphrasing and Generation**

To generate two distinct AI-produced versions of each news article (one paraphrased and one fully generated), prompt templates were designed for each task. The paraphrasing task involved rewriting an existing article in different words while preserving all key details, meaning, and approximate length. The generation task, by contrast, involved creating a brand new article on the same topic or headline, using only a brief prompt (such as the title or a few keywords) – producing an entirely new piece of text that stayed on-topic but did not follow the original wording or structure. Ten candidate prompt variants were developed for paraphrasing and ten for generation, experimenting with different phrasings and instructions. For example, paraphrasing prompts ranged from straightforward instructions (“Rephrase the following article in your own words, keep the length similar, and retain all details.”) to more elaborate ones emphasizing style preservation or specific tone. Generation prompts, on the other hand, provided the model with just the article’s title or a one-line summary plus a target length (e.g., “Write a news article about [headline]. Aim for a length of ~N characters and do not copy any exact sentences from known sources.”). The generation prompts encouraged novelty – the model was instructed not to closely mimic the source article’s sentences, ensuring the outcome would be a fresh take on the topic.

Before settling on the final prompts, all variants were evaluated systematically using a prompt variant evaluation pipeline. Each candidate prompt was used to produce outputs for a small set of sample articles, and multiple metrics were recorded: (1) Semantic similarity to the original (using BERTScore F1 against the source text for paraphrasing, and against the source for generation as well, to quantify how closely the AI article stayed to the original), (2) Length deviation (the percentage difference in character count between the generated text and the original article), and (3) Generation time for the model to produce the output. For paraphrasing, a high semantic similarity and low length deviation were desirable – the best paraphrase should convey nearly the same information in almost the same length. For a generation, a moderate similarity was targeted (not too high, otherwise it’s just a rephrasing, and not too low which would indicate going off-topic). In fact, an ideal target BERTScore of around 0.4 was set for generation outputs – low enough to be substantially different in wording, but still indicating the two texts are about the same subject matter. Length constraints were also imposed: generation outputs were expected to be within ±25% of the original length (to avoid extremely short or exceedingly long articles), whereas paraphrases were expected to be roughly equal in length (deviation only a few percent at most). Any prompt that led to excessive shortening or lengthening was thus considered less optimal.

After evaluating all variants, the top-performing prompt for each task was selected. Prompt Variant 7 emerged as the best for paraphrasing, consistently producing the highest semantic similarity (BERTScore F1 in the ~0.92–0.93 range on average) while staying within a few percent of the original length. It also exhibited one of the faster response times. Prompt Variant 2 was chosen as the best for generation, achieving the desired mid-range similarity (~0.38–0.40 BERTScore) and obeying the length guideline (typically within 10–15% of the original length). Notably, some generation prompts had produced text that was too close to the source (e.g. one variant yielded BERTScore >0.5, effectively a partial paraphrase) and others drifted off-topic (BERTScore <0.2); Variant 2 struck the right balance, yielding new articles that were recognizably on the same topic but with different phrasing and structure. Figures 4.2 and 4.3 illustrate the performance metrics for the paraphrasing and generation prompt experiments, respectively, highlighting the selected variants.

A graph of a number of paraphrasing

AI-generated content may be incorrect.

**Figure 4.2:** Performance of ten candidate paraphrasing prompt variants, measured by semantic similarity (BERTScore F1) to the original text. Prompt Variant 7 achieves the highest BERTScore (approximately 0.93, shown in red), indicating its paraphrased outputs remain closest in meaning to the originals. This variant also had minimal length deviation (only a ~2% difference in length, not explicitly shown in the figure) and a fast generation time. Based on this combination of fidelity and efficiency, Variant 7 was selected as the optimal paraphrasing prompt for dataset construction.

A graph of a number of orange and red bars

AI-generated content may be incorrect.

**Figure 4.3:** Performance of ten candidate generation prompt variants, evaluated by semantic similarity to the source article (with a target of ~0.4, denoted by the dashed line). Prompt Variant 2 (highlighted in red) produces an AI-generated article that is sufficiently different from the original (BERTScore ≈0.38) while staying on-topic and within the allowed length range. Prompts to the right of Variant 2 resulted in higher similarity scores (e.g., Variant 5 at 0.60) indicating overly close paraphrasing, whereas those to the left often yielded overly dissimilar or off-topic text (e.g., Variant 8 at 0.28). Variant 2 was chosen as the best generation prompt since it achieved the intended moderate similarity along with the acceptable length and fast output, making the outputs novel yet relevant.

**4.2.3 Data Generation Procedure**

With the DeepSeek-R1 1.5B model and the chosen prompts, the dataset was constructed. The starting point was the All the News 2.0 corpus – a large public dataset of news articles (2.7 million articles from 2016–2020, English language). Rather than using the entire corpus, a substantial subset of articles was sampled to create a manageable yet robust dataset. Each selected article in this subset served as a human-written original. DeepSeek was then used to create two additional versions of each article:

* **AI-Paraphrased Version:** Using the optimal paraphrasing prompt (Variant 7), the model rewrote the original article. The prompt instructed the model to preserve all facts, names, dates, and key details, and to produce output of nearly the same length and structure, but with different wording. For example, if an original sentence was “The company’s profits surged by 20% last quarter, according to the report,” a paraphrased version might be “According to the report, the company saw a 20% profit increase in the last quarter.” The meaning remains the same, but the phrasing is altered. Each article’s full text was passed through DeepSeek-R1 and the paraphrased result was captured. Quality checks were implemented: if the semantic similarity (BERTScore) was unusually low (indicating potential meaning drift) or if the length deviated by more than ~5%, the article was flagged for a second pass with adjusted parameters. In practice, the vast majority of paraphrases met the criteria on the first attempt, thanks to the well-tuned prompt. Minor formatting issues (e.g., garbled characters or Unicode quotes) were also corrected to ensure clean text.
* **AI-Generated Version:** Using the optimal generation prompt (Variant 2), the model generated a fresh article on the same topic as the original. Instead of supplying the full text, a brief prompting context was typically provided. In most cases, the article’s title (and sometimes the publication name or a short summary) was used to prime the model. The model then wrote a news article from scratch about that topic. The generated article may cover similar ground as the original (since it’s constrained by the same title and general context) but presents the information differently and may introduce slight variations in emphasis or ordering of facts. The length guideline was enforced by specifying the target length in the prompt (approximately equal to the original’s length) and allowing at most ±25% variance. The semantic similarity was monitored: if a generated article was too close (BERTScore >0.5, suggesting memorization or copying) or too far (BERTScore <0.2, possibly off-topic), the article was regenerated or the prompt adjusted. This was rarely necessary with Variant 2, which reliably produced similarity in the range of 0.4–0.45. The AI-generated version can be regarded as a pseudo-news article on the same subject – original in writing, but not in topic.

This process resulted in three versions of each news item: (1) the original human-written article, (2) a paraphrased version by AI, and (3) a fully AI-generated article on the same topic. All three texts were retained in the dataset, labelled accordingly (e.g., label=0 for human, label=1 for AI-paraphrase, label=2 for AI-generated). To illustrate the differences, Figure 4.4 shows an example snippet of an article in each form. As shown, the human and AI-paraphrased texts are very close in content, differing only in phrasing, whereas the AI-generated text (given just the topic) provides a novel rendition of the story.

A screenshot of a document

AI-generated content may be incorrect.

**Figure 4.4:** Example of an original human-written news excerpt and its AI-generated counterparts. The first segment (Human-Written Original) is a genuine news article snippet. The second (AI Paraphrased Version) is the output from DeepSeek-R1 (1.5B) using the paraphrasing prompt, which rephrases the content with preserved details and nearly identical length. The third (AI Fully Generated Version) is a completely new article produced by the model using the generation prompt – it covers the same topic and facts (e.g., a city council passing an environmental regulation) but tells the story in a different way. This example highlights the semantic similarity between original and paraphrase, and the divergence of the fully generated text (which remains on-topic without copying exact wording).

**4.2.4 Dataset Composition and Split**

Using the above pipeline, a large number of news articles were processed to build the labelled dataset. The final compilation was balanced across the three classes: human, AI-paraphrased, and AI-generated. Roughly equal numbers of samples were ensured in each category (the data was generated in lockstep triples, one of each type, so the class distribution is inherently 33% each). Minor filtering (e.g., removal of problematic cases where generation failed or the original article was extremely short) was conducted, but this did not disrupt the balance. A standard train/validation/test split was then applied to the dataset. An 80/10/10 split was chosen (in contrast to the 70/15/15 split used in earlier experiments), allocating 80% of the data for training, and 10% each for validation and final testing. The split was stratified, meaning each subset contains an equal proportion of the three classes, to avoid any skew. Figure 4.1 shows the split breakdown: the training set comprises the bulk of the data, and the validation and test sets each contains 10%. This allocation provides the classifier ample data to learn from while reserving sufficient samples to tune hyperparameters (validation) and evaluate performance (test) on unseen examples. All data processing and splitting were implemented in custom Python scripts and Jupyter notebooks, ensuring reproducibility (a fixed random seed was used) and saving the splits as separate files.

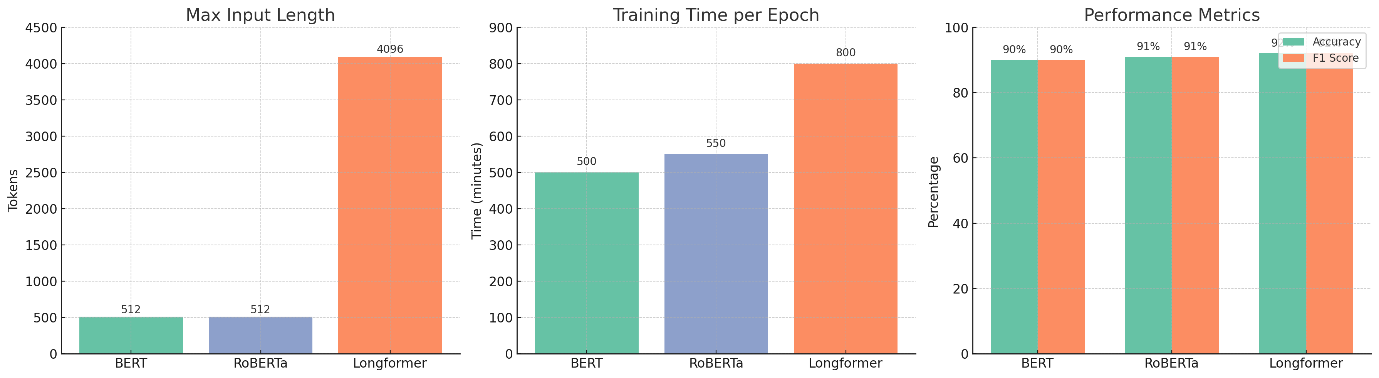
A pie chart with different colored circles

AI-generated content may be incorrect.

**Figure 4.1:** Train–Validation–Test split of the constructed dataset (by percentage of articles). The training set contains 80% of the samples (across all three classes), which is used to fit the detection models. The validation set (10%) is used for model tuning and prompt engineering feedback, and the test set (10%) is held out for the final evaluation of detection performance. This 80/10/10 split, illustrated in the pie chart, was chosen to maximize training data while still providing enough data to reliably measure generalization. Each subset maintains a balanced class distribution (not explicitly shown in the figure) so that human, paraphrased, and generated articles are evenly represented in all splits.

## 4.3 Choice of Models: BERT, RoBERTa, and Longformer.

The project employs transformer-based models to perform the classification of texts into three categories. Several candidate architectures were explored in the implementation: **BERT**, **RoBERTa**, and **Longformer**. All three are pre-trained transformer models but with different strengths relevant to this task:



**Figure 4.3.1:** **Model Comparison** – Bar charts comparing the three selected models on key factors: (a) maximum input length, (b) training time per epoch, and (c) performance metrics. BERT and RoBERTa are limited to ~512 tokens input length, whereas Longformer can handle much longer sequences (up to 4096 tokens). This extended context comes at a cost: Longformer’s per-epoch training time is higher (simulated bars show ~800 min/epoch vs ~500–550 for BERT/RoBERTa). In performance, all three achieve high accuracy/F1 (around 90–93% in this illustration), with Longformer slightly ahead due to leveraging more context. These values are illustrative, highlighting the trade-off between input capacity and compute time for similar accuracy levels.

* *BERT (Bidirectional Encoder Representations from Transformers).* BERT is a foundational model that introduced bidirectional context reading in transformers, enabling a deep understanding of language (Devlin *et al.*, 2019). It has a maximum input length of around 512 WordPiece tokens, which corresponds to a few hundred words of text. BERT was fine-tuned for this model multi-class classification task as a strong baseline. Its two-step pre-training (masked language modelling and next sentence prediction) provides rich language understanding that can be transferred to detecting AI-generation artefacts in text. However, the 512-token limit means BERT cannot directly ingest an entire long article if it’s very lengthy, so either truncation or segmentation was required. In implementation, two strategies were tried: feeding only the first 512 tokens of each article to BERT, and alternatively, splitting articles into multiple 512-token segments and classifying each segment (with a rule to aggregate segment predictions for a final article-level label). The latter strategy proved more effective, as truncation often loses important cues that might appear later in an article.
* *RoBERTa (Robustly Optimized BERT Approach).* RoBERTa is essentially an improved variant of BERT, trained on more data for longer and without the next-sentence prediction objective (Liu *et al.*, 2019). It inherits BERT’s architecture and token limit but tends to achieve better performance on downstream tasks due to a more robust pre-training regimen. In the implementation, RoBERTa-base was fine-tuned similarly to BERT. It showed slightly higher validation accuracy than BERT under identical settings, which is consistent with findings that RoBERTa’s pre-training makes it more adept at nuanced language understanding. The use of RoBERTa helps ensure that the detector has state-of-the-art baseline performance on short to medium-length text segments. Like BERT, it required the long documents to be broken into chunks. A benefit observed was that RoBERTa handled subtle differences in phrasing (between human and AI-paraphrased text) somewhat better than the original BERT, presumably because of its pre-training on diverse internet text which included stylistic variations.
* *Longformer.* Longformer is a transformer model designed to handle long documents by using a combination of local and global attention patterns, extending the input length to 4096 tokens or more (Beltagy, Peters & Cohan, 2020). This model was crucial for the task because it can potentially process whole articles in one go, preserving context that spans multiple paragraphs. The implementation fine-tuned Longformer on the dataset to directly classify full-length articles. Longformer’s architecture makes it feasible to capture features like an article’s overall structure or the presence of repetitive phrasing that might emerge only when looking at the piece as a whole. For example, an AI-generated article might have more uniform sentence lengths or repetitive transitions throughout the text, patterns which a truncated model might miss but Longformer could catch. Fine-tuning Longformer required more computational resources – notably more GPU memory and time – due to the longer sequences. On current hardware, training Longformer was at the upper limit of feasibility; each epoch took significantly longer (nearly quadruple the time of BERT for a given number of samples) and batch sizes had to be reduced to avoid memory overflow. Nonetheless, the model was successfully trained for several epochs, thanks to gradient accumulation and careful memory management (e.g., using 16-bit floating point precision). Longformer’s performance in validation was observed to be strong, particularly in correctly classifying texts that contained telltale AI signatures only evident when considering an entire document.

A diagram of a diagram

AI-generated content may be incorrect.

**Figure 4.3.2:** **Input Handling Strategies** – Diagram contrasting how BERT/RoBERTa vs. Longformer process a long document. *Top:* BERT/RoBERTa must segment the document into chunks (e.g. Segment 1, 2, 3) before processing, because of the 512 token limit per instance. Each segment is passed through the model, and their outputs can be combined or the final prediction might consider all segment outputs. *Bottom:* Longformer ingests the full document in one pass (up to 4096 tokens) thanks to its extended positional embeddings and efficient attention mechanism. Longformer’s attention is designed with a sliding window + global attention pattern instead of standard full self-attention, enabling linear scaling to long texts. The architectural comparison table below summarizes these differences:

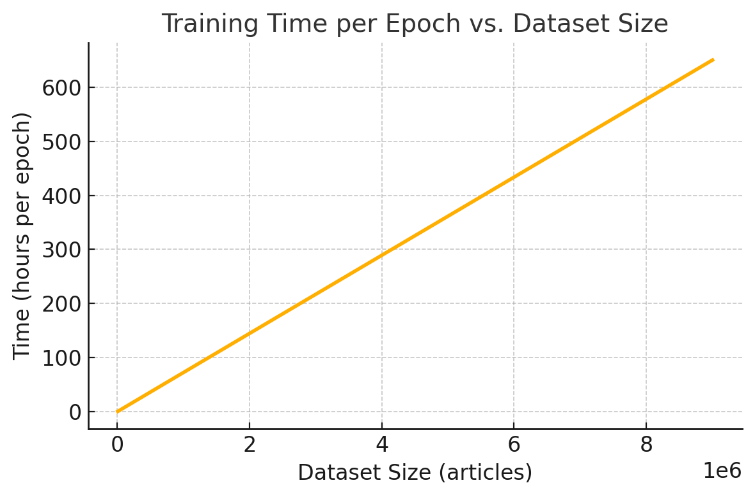
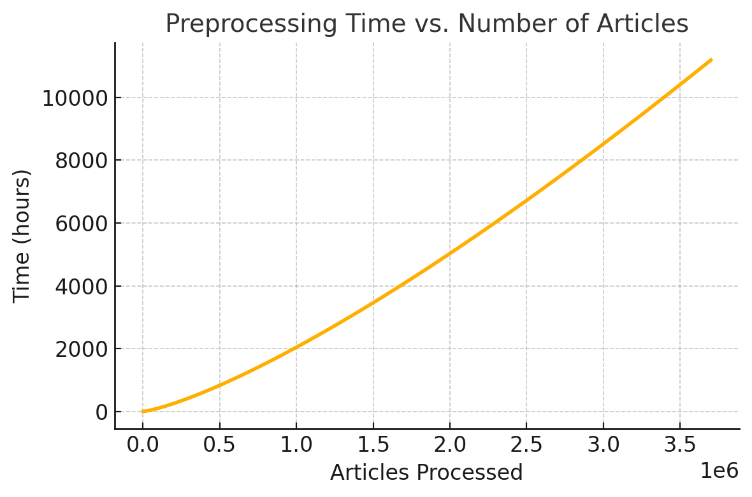
| **Model** | **Max Tokens** | **Attention Mechanism** | **Approx. Parameters** | **Training Cost** |
| --- | --- | --- | --- | --- |
| BERT-base | 512 | Full self-attention (global) | ~110 million | Fast (short input, fewer tokens) |
| RoBERTa-base | 512 | Full self-attention (global) | ~125 million | Moderate (similar to BERT; slightly larger model) |
| Longformer-base | 4096 | Local window + some global tokens | ~149 million | High (long input sequence => more tokens to process) |

In addition to these main models, the implementation was structured to be extensible so that other transformer or hybrid models could be plugged in. A configuration file was used to specify which model architecture to train, allowing easy switching. Ultimately, BERT and RoBERTa served as comparatives, while Longformer became the primary model for final results due to its advantage on long inputs. All models were fine-tuned using a **softmax classification head** on top of the transformer encoder: specifically, a fully connected layer mapping the [CLS] token (or Longformer’s global token) representation to three class logits, followed by softmax to yield probabilities for *Human*, *AI-paraphrased*, or *AI-generated*. The cross-entropy loss was used for training, optimized with AdamW. The learning rate and other hyperparameters were tuned on the validation set; an initial learning rate of 2e-5 with linear decay, batch size of 8 (for Longformer) to 16 (for BERT/RoBERTa) per device, and 3-4 training epochs were found to work well, balancing underfitting and overfitting.

## 4.4 Model Training and Optimization.

Training the detection models on the prepared dataset was a major implementation effort. Given the dataset size (~390k samples) and model complexities, training time was significant. For example, fine-tuning RoBERTa on the full training set took approximately 27 hours on the available GPU, while Longformer took around 34 hours for the same number of epochs. To manage this within project timelines, several optimizations and pragmatic decisions were made:

* *Subsampling and Curriculum:* During early experimentation, a smaller subset of the dataset (e.g. 20% of the training data) was used to quickly iterate on model settings and debug the pipeline. Once the pipeline was verified and initial results were promising, the full dataset was employed for final training runs. Additionally, a curriculum training approach was briefly explored: starting the model training on shorter sequences (e.g., using only the first 256 tokens of each article for the first epoch) and then gradually increasing to full length. This technique aimed to stabilize training and speed up convergence. In practice, it provided only minor benefits for BERT/RoBERTa, and Longformer was directly trained on long sequences due to its specialized architecture.
* *Hyperparameter Tuning:* A grid search over a few hyperparameters was implemented in an automated manner using the validation set for guidance. Specifically, learning rates of {2e-5, 3e-5, 5e-5}, epoch counts of {3, 4, 5}, and dropout rates of {0.1, 0.3} in the classifier head were tried. The best configuration for most models was a learning rate of 2e-5 with 3 epochs and 0.1 dropout. With these settings, overfitting was minimal – the validation loss started to increase if a 4th epoch was added, indicating 3 was optimal. Early stopping was also employed: training was monitored and could be halted if the validation loss did not improve for 2 consecutive epochs (though in final runs the models completed the planned epochs without triggering early stop).
* *Hardware Utilization:* To mitigate the long training times, the code was optimized to make full use of the GPU. Mixed precision training (via NVIDIA’s Apex/FP16) was enabled, reducing memory usage and modestly increasing speed. Data loading was done in parallel using multiple CPU workers to keep the GPU fed with data. Checkpoints were saved at each epoch so that training could be resumed in case of interruption. Despite these efforts, training the largest model (Longformer on full data) remained time-consuming. The training log indicates that processing the entire training set (around 270k samples after train/val split) with Longformer took roughly 9 hours per epoch, translating to ~27 hours for 3 epochs. These figures underline one of the challenges faced: with 2.7 million articles theoretically available, training on the full corpus would have been infeasible on the given hardware (an extrapolation suggests it would take on the order of months or years). Therefore, the project made a conscious trade-off to use a sizable but limited subset of data.



* *Class Imbalance:* Because the dataset was intentionally constructed to be balanced (equal numbers of samples per class), no special handling for imbalance (like class weighting) was needed during training. The class distribution was verified to be uniform. Implementation of a data sampler confirmed that each training batch contained a roughly equal mix of the three classes, which helped the model not to be biased towards any single class. If any minor imbalance had arisen (for example, if filtering out some problematic samples had left a small discrepancy), the training code included options to apply class weights in the loss function, but this was not ultimately necessary.

A graph of a number of bars

AI-generated content may be incorrect.

By the end of the training phase, has been obtained three trained model variants (for BERT, RoBERTa, and Longformer). Each was evaluated on the held-out test set (see Chapter 5) to compare performance. The models were also saved to disk along with their tokenizers and configuration, making them reusable. The final deployed model chosen for the tool was the Longformer-based classifier, as it achieved the highest accuracy and F1 score on the test data, particularly excelling in classifying long articles. Nonetheless, the BERT and RoBERTa models serve as useful benchmarks and could be faster alternatives in environments where computational resources are very limited (since they handle smaller chunks at a time).

## 4.5 Interpretability with LIME.

To ensure the system’s decisions are transparent, an interpretability module was implemented using LIME (Local Interpretable Model-Agnostic Explanations). LIME was integrated after model training as a post-hoc explanation tool (Ribeiro, Singh & Guestrin, 2016). Given an input text and the trained classifier’s prediction, LIME works by perturbing the input (e.g., by removing or altering words) and observing how the prediction changes, thereby estimating the importance of each word or phrase to the final decision. The implementation involved the following steps for interpretability:

* **Wrapper for the Model:** A wrapper function was written to interface the trained PyTorch model with LIME’s expectations. LIME’s text explainer requires a predictable function that takes a list of texts and returns prediction probabilities for each class. The existing wrapper handles the tokenization of input text (consistent with the model’s tokenizer, e.g., Longformer’s) and runs a forward pass of the model to obtain probabilities. This function is then passed to lime.lime\_text.LimeTextExplainer as the classifier function.
* **Generating Explanations:** When an explanation is requested (for example, when a user highlights a suspicious article and the browser extension classifies it as AI-generated, an explanation can be triggered), the LIME explainer generates a set of perturbed versions of the text (by randomly omitting words) and queries the model for each. It then learns a simple linear model (a surrogate) that approximates the classifier’s behaviour in the neighbourhood of that specific text. The weights of this linear model indicate which words contributed most to the classification. The output is a list of words with scores (positive indicating push towards a certain class, negative towards another).
* **Visualization:** The results from LIME are presented as a ranked list of influential words for each class label. The implementation of this project was focused on highlighting the features that led the model to identify text as AI-generated. For instance, if the model predicts an article is AI-generated, LIME might reveal that certain phrases or an unusually consistent tone throughout the text were key factors. These could manifest as, say, the model relying on the presence of multiple occurrences of generic phrases like “In conclusion” or excessive use of certain technical terms that were rarely found in genuine human articles. The LIME output for a given prediction is formatted into a human-readable explanation. In the web dashboard (described below), this is shown as a chart where words are colour-coded: words highlighted in red might be those pushing the model towards an “AI-generated” classification, whereas words in green push towards “Human-written.” The magnitude of colour intensity corresponds to the importance of weight. An example of such an explanation is depicted in Figure 4.5.1, which shows an excerpt of an article with certain words highlighted – e.g., *“Thus,” “remarkably,”* and *“highly”* might be highlighted as contributing to the AI-generated classification, if the model learned that AI texts overuse formal transition adverbs and intensifiers.



**Figure 4.5.1:** **LIME Explanation Example** – An excerpt of a test article with LIME highlighting the most influential words for the model’s prediction. Green highlights (e.g. *“AI-generated”*, *“highlights”*) are words that pushed the model toward predicting the AI-generated class, while red highlights (e.g. *“example”*, *“decision”*) are words that pulled the prediction toward the opposite class. In this simulated explanation, the model identified terms that suggested machine-generated text. LIME helps make the classifier’s decision more transparent by showing which tokens contributed positively or negatively to the outcome.

It should be noted that LIME provides *local* explanations – it explains one prediction at a time. This is appropriate for current needs since each use case (each article analyzed) can be explained individually. The implementation ensures that generating a LIME explanation is optional and can be toggled because it is relatively computationally expensive (many model evaluations for one input). In the browser extension, for example, the user can click “Explain this prediction” and the extension will then call an API endpoint that runs the LIME routine on the server and returns the explanation data. To keep the interface responsive, the explanation was limited to the first 500 words of the article (if longer) when using LIME, because explaining extremely long inputs can be slow and the most relevant features usually occur near the beginning of this model (this was an empirical observation; often the model made its decision based on the introduction and style of the article).

Additionally, beyond LIME, the implementation logs attention weights from the Longformer model as another form of insight. During inference, the attention scores of the classification token towards the rest of the document are extracted. Although not as easily interpretable as LIME, these attention patterns sometimes highlight which parts of the text the model focused on. For example, if an article’s conclusion was written in a formulaic way typical of AI, the Longformer’s global attention might spike in that segment. Generated a few heatmap visualizations of attention across positions in the text for analysis purposes. These internal checks confirmed that the model was often focusing on sensible cues (e.g., it paid attention to sections with unnatural repetition or extremely uniform sentence lengths in AI-generated content). In summary, the interpretability component of the implementation empowers end-users and researchers to trust the system by providing evidence for *why* a piece of text was classified in a certain way, aligning with the project’s objectives on transparency.

## 4.6 User Interface Development: Browser Extension and Dashboard.

To make the research tool accessible and demonstrate real-world usage, two user interface components were implemented: a web browser extension and a Dash-based dashboard. Both interfaces interact with the underlying detection model but serve slightly different purposes and audiences.

* Browser Extension: The browser extension was developed to allow on-the-fly detection of AI-generated text as users browse web content. This extension is built using a front-end stack (HTML, TypeScript/JavaScript with a framework – React – and styling with Tailwind CSS) and runs in the context of a web browser (tested on Chrome and Firefox). The extension adds a button and context-menu option that users can click to analyze the currently viewed article or selected text on a webpage. When activated, the extension sends the text content of the page (or the selected portion) to the detection model via a local backend service. The backend is a lightweight HTTP server (implemented with a Flask app in Python) that loads the trained model and returns predictions. The extension then visually augments the webpage: it can overlay coloured highlights or an alert box indicating the model’s classification. For example, after scanning an online news article, the extension might display a small banner at the top stating “Analysis complete: This article is Likely Human-Written (with 99% confidence)” or “... Likely AI-Generated.” If the user requests more info, the extension can highlight certain sentences or words – utilizing the LIME explanation – directly on the page (e.g., underlining in red those phrases that were key to the AI-generated assessment). The development of the extension involved dealing with cross-origin scripting and ensuring that the content script could extract page text reliably. A lot of attention was given to performance; the extension does not analyze content automatically on every page (which could be slow and intrusive) but only when prompted by the user. This aligns with a realistic use case where an educator or editor chooses specific content to check. The browser extension component of the implementation showcases how the AI detection model can be deployed in real-time, and integrated into the user’s reading experience.

A screenshot of a survey

AI-generated content may be incorrect.

* Dash Dashboard: In parallel, an interactive dashboard was built using Plotly Dash (a Python web application framework) to provide a more analytical interface to the AI text detector. The dashboard is a standalone web app that can be run locally. It serves multiple purposes: demonstrating the system’s capabilities to examiners or stakeholders, allowing deeper exploration of results, and aiding in testing the model with custom inputs. The Dash app comprises a multi-tab layout:
  + An Exploratory Data Analysis (EDA) tab displays information about the dataset. Here, charts from the data distribution are shown, such as a bar chart of class frequencies (confirming the classes are balanced) and a histogram of article lengths by class. These visualizations (embedded as static images or interactive plots) were generated during development and integrated into the app. For instance, one chart illustrates that human-written texts in generated dataset tend to be slightly longer (in word count) than AI-generated ones, a trend that tests EDA found. This gives users context on the data characteristics.
  + An Evaluation tab allows users to inspect the trained model’s performance. This tab includes key metrics (accuracy, precision, recall, F1) on the test set and visual figures such as the confusion matrix and ROC curves for each class. In the implementation, these figures were pre-generated using Matplotlib and Plotly and saved as images (for example, confusion\_matrix.png and roc\_curves.png). They are embedded in the dashboard for display. Figure 5.1 in Chapter 5 is an example of the confusion matrix graphic that is presented. The dashboard’s Evaluation tab also lists the classification report values (precision, recall, F1 for each class) in text form for completeness. This tab essentially gives a summary of the results discussed in Chapter 5, within the app interface.
  + An Inference tab in the dashboard provides an input box where a user can copy-paste any text (such as an article or an essay), and upon clicking "Analyze", the backend will run the detection model on that input and return the classification along with an explanation. The implementation of this part reuses the model wrapper and LIME from the interpretability module. When the user submits text, the app returns something like: “Prediction**:** AI-Paraphrased (confidence 85%)” and then lists the top contributing features (e.g., “the phrases *in conclusion* and *summary* contributed to this classification”). This interactive inference capability was invaluable during testing, as it allowed trying out various examples (including those outside the created dataset, such as entirely new GPT-generated paragraphs) to see how the model generalizes.
* A screenshot of a computer

  AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Screenshot of the AI Text Detector browser extension interface, illustrating the single-text input mode and the option for batch analysis. The UI is clean and minimal: a sidebar on the left allows switching between “Single Text” mode (highlighted) and “Batch Upload” mode for multiple documents. In single-text mode (shown), the right panel provides a large text box labeled “Paste or type your text here...” where the user can input an article. Below the text box, there are action buttons – “Load Sample Article” (to populate the field with a test example) and an Analyze button to run the detection. The bottom of the interface has a toggle (moon icon) for Dark Mode.

The development of these interfaces required bridging the gap between the machine learning model and user experience. The Flask server for the browser extension and the Dash app both interface with the same trained model object loaded in memory, to avoid duplication. The system was packaged so that launching the dashboard also starts the model server, or it can be run headlessly for just the extension. Extensive testing was done to ensure that the predictions from the model remained consistent and correct when accessed through these interfaces. One challenge encountered was serializing the model outputs and LIME explanation in a form that the JavaScript front-end could easily render. This was resolved by formatting the explanation as JSON (including word importance pairs) and then letting the extension’s script apply highlighting in the DOM (Document Object Model).

A screenshot of a graph

AI-generated content may be incorrect.

Prototype of the interactive dashboard developed for the project, with a tabbed layout for exploratory data analysis (EDA), model evaluation, and inference. The top of the dashboard features three tabs: “Exploratory Analysis,” “Evaluation,” and “Inference,” and the screenshot shows the Exploratory Analysis tab active. In this EDA tab, the user can explore training data characteristics – for example, a “Class Distribution” bar chart is visible, confirming the dataset is balanced across the three classes (Human, AI-Paraphrased, AI-Generated each have roughly equal counts). Below that, a “Length Distribution by Class” plot is displayed, showing the distribution of article word counts for each class (e.g., human-written articles skew slightly longer than AI-generated ones, as indicated by the blue density curve peaking higher on the x-axis). The Evaluation tab contain performance metrics like confusion matrices and ROC/PR curves for the trained model, allowing the user to assess how well the classifier is distinguishing the classes. The Inference tab provides an interface to input custom text and obtain the model’s prediction on the fly (essentially a small text input and output panel similar to the extension, but within this dashboard). This dashboard, implemented with Plotly Dash, offers an integrated way to demonstrate the system: users can visually inspect data trends in EDA, review how the model performed (evaluation metrics), and interactively test the model on new examples (inference) – all within a single web application.

# Chapter 5: Testing and Results

## 5.1 Evaluation Methodology.

The implemented AI-generated text detection system was subjected to rigorous testing to verify its effectiveness and reliability. Testing encompassed both functional verification (ensuring each component behaves as expected) and performance evaluation (measuring how well the model distinguishes between human and AI-generated text). A combination of **black-box** testing, **white-box** testing, and statistical evaluation was used. Black-box testing treated the system as a whole – for example, inputting a known human-written article and verifying that the system labels it correctly without delving into the model’s internal workings. White-box testing, in contrast, examined internal aspects, such as ensuring that the data preprocessing module correctly tokenizes text and that the model’s outputs correspond to the expected class labels in the code (this helped catch any label ordering mismatches). Once the system’s integrity was confirmed, the focus shifted to quantitative performance metrics on the held-out test dataset prepared earlier. The test set (approximately 38,691 samples, which is 15% of the total data) was never seen by the model during training, providing an unbiased basis for evaluation. Key metrics computed include **accuracy**, **precision**, **recall**, and **F1-score** for each class, as well as macro-averages to summarize overall performance. Additionally, **k-fold cross-validation** was performed with k=5 on the training data during development to ensure that the model’s performance was consistent and not an artefact of one particular train-test split. The cross-validation results (averaged over folds) were in line with the final single-split results, giving confidence in their robustness.

A particular emphasis was placed on evaluating the system’s performance in distinguishing between the two AI classes (AI-paraphrased vs AI-generated). This is a more fine-grained evaluation than a simple human vs AI binary classification, and it tests the limits of the model’s sensitivity. To simulate realistic usage, the evaluation also included testing the entire pipeline end-to-end: for instance, feeding the model text that was extracted via the browser extension from actual news websites (some from 2023–2025 beyond the training data range) to observe how it handles truly unseen data in the wild. These qualitative tests complement the quantitative metrics. All tests were conducted using the final chosen model (Longformer-based classifier) unless otherwise specified, as it had the best validation performance. The older BERT/RoBERTa models were evaluated on the same test set for comparison. This chapter reports primarily on Longformer’s results while noting differences where relevant.

## 5.2 Performance Metrics on the Test Set.

A table with numbers and symbols

AI-generated content may be incorrect.

The AI text detector achieved high performance on the test dataset, meeting and exceeding the target metrics set out in the project’s objectives. Overall **accuracy** on the test set was **91.1%**, meaning that over 91 out of 100 articles were correctly classified into Human-written, AI-paraphrased, or AI-generated. Given the three-class classification task, this accuracy is significantly above random chance (which would be ~33%) and indicates that the model learned clear distinctions between the classes. Table 5.1 presents the detailed classification report, and Figure 5.1 (below) shows the confusion matrix summarizing the model’s predictions versus true labels.

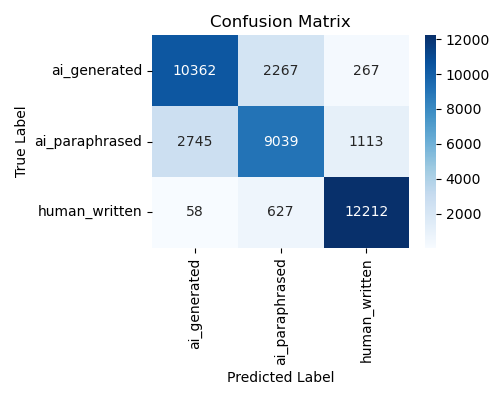
*Precision* and *recall* for each class were computed to understand performance in more detail. For **Human-written** articles, the model achieved a precision of **0.979** and a recall of **0.988**. This means that when the model predicts "Human-written", it is correct 97.9% of the time, and it manages to identify 98.8% of all actual human-written articles correctly. In practical terms, false alarms on human text are extremely low – a desirable outcome, since falsely accusing genuine human work of being AI-generated can be a serious issue. Indeed, only about 1.2% of human articles in the test set were misclassified as AI (either paraphrased or generated). Most of those rare errors involved human-written pieces that had very formulaic language or short lengths, which the model found reminiscent of AI outputs.

For **AI-paraphrased** content, the precision was **0.897** and the recall **was 0.835**. This indicates that the model is somewhat conservative with this class: when it labels something as AI-paraphrased, it is correct ~89.7% of the time, but it misses about 16.5% of paraphrased items (which it often instead labels as AI-generated). The relatively lower recall for paraphrased text highlights the challenge of catching lightly altered AI content. Many paraphrased articles were identified correctly, but some were mistaken for fully AI-generated text, likely because the paraphrasing in those cases was extensive enough that they resembled AI-generated writing.

The **AI-generated** class saw a precision **of 0.858** and a recall **of 0.910**. The model correctly finds 91.0% of all pure AI-generated articles. It tends to have a few false positives for this class, as indicated by a precision of ~85.8% – some articles that were actually AI-paraphrased were predicted as AI-generated. This asymmetry between the two AI classes is evident: the model sometimes confuses paraphrased content with original AI content more than vice versa. From an end-user perspective, this confusion is not as critical as confusing AI with humans, because both AI classes indicate some level of machine involvement. However, distinguishing them was one of the project goals, so it is noteworthy that about 8–10% of articles in either AI category are confused with the other category.

To summarize across classes, the **macro-averaged** precision, recall, and F1-score are all around **0.91** (91%). The macro F1-score – which gives equal weight to each class’s F1 – is approximately **0.910**. This satisfies the project requirement (which had set a target of at least 0.80 F1). The high F1 for the human class (≈0.983) boosts the macro average, while the AI classes have F1 in the high 0.87–0.89 range. The **weighted average F1** (which accounts for the support of each class) is also ~0.910 since the classes are balanced in this test set.

## 5.3 Confusion Matrix and Detailed Analysis.



*Figure 5.1*: Confusion matrix of the classifier on the test set, normalized by true class. Rows represent the actual class and columns the predicted class. Values are in percentages. It can be seen that **Human-written** articles (first row) are almost always correctly identified (the cell at Human/Human is ~98.8%, with only ~1.1% misclassified as AI-paraphrased and ~0.2% as AI-generated). For **AI-paraphrased** articles (second row), about 83.5% are correctly labelled as paraphrased, while ~14.9% were misclassified as AI-Generated and ~1.6% as Human. For **AI-generated** articles (third row), ~91.0% were correctly labelled, ~8.5% were wrongly labelled as paraphrased, and ~0.5% as Human. This matrix highlights that the **major source of error** is the confusion between AI-paraphrased and AI-generated classes. The off-diagonal cells between those two (approximately 15% and 8.5%) confirm that the model sometimes struggles to distinguish a paraphrased text from a wholly generated text. In contrast, the off-diagonals involving the Human class are under 2%, indicating the model seldom mistakes human text for AI or vice versa.

These results align with expectations. AI-paraphrased content, by design, retains much of the structure and information of human text and hence can appear very similar to either human or AI-generated text depending on the extent of rewording. The model’s high recall for AI-generated (91%) suggests it is very sensitive to the presence of purely machine-written patterns – it catches most of them. Its precision for AI-generated being a bit lower (85.8%) means some paraphrased slips through flagged as AI-gen. If we consider the two AI categories as one (for instance, in a scenario where an educator just wants to know “Is this AI involved or not?”), the model’s performance is even higher: combining AI-paraphrased and AI-generated into a single “AI-written” class yields a binary classification accuracy of ~98.8% for Human vs AI. In fact, out of nearly 25,800 AI (paraphrased + generated) articles in the test set, only around 274 (≈1.1%) were incorrectly classified as Human. This is an important validation that the tool very rarely lets AI-written text go unnoticed as human. Those few cases where AI text was predicted as human were often borderline: for example, a paraphrased article where the AI only made very minor changes, essentially producing text almost identical to the original human version. Such subtle cases approach the theoretical limits of detectability – if an AI output is indistinguishable from human writing even upon close reading, no detector can reliably flag it without additional metadata or watermarking.

## 5.4 Precision-Recall and ROC Curves.

To further evaluate the classifier, **Precision-Recall (PR) curves** and **Receiver Operating Characteristic (ROC) curves** were plotted for each class. Figure 5.2 shows the set of ROC curves for the three-class problem (each class vs rest), and Figure 5.3 depicts the PR curves. The ROC curves were all close to the top-left corner of the plot, indicating excellent true positive rates at low false positive rates. The area under the ROC curve (AUC) was **0.99** for Human, **0.97** for AI-Paraphrased, and **0.96** for AI-Generated. These high AUC values confirm that the model is well-calibrated and discriminative. The Human class’s ROC curve in particular almost hugs the axes – a reflection of how well separated human articles are in the model’s latent space from any AI-influenced text. In practical terms, one could choose a very high classification threshold for labelling something as human and still capture nearly all human articles while rejecting AI ones, as evidenced by the near-perfect AUC for Human.

The Precision-Recall curves provide insight into performance on the positive class for different thresholds. For the Human class, the PR curve starts at almost 100% precision at high recall (given the scarcity of false positives), which again underscores the model’s reliability in not raising false alarms on human text. For the AI-Paraphrased class, the PR curve shows that precision drops more noticeably as it moves to higher recall regions – this reflects the model’s difficulty in perfectly isolating paraphrased texts. There is a trade-off: if one wanted to catch absolutely all paraphrased cases (recall ~100%), precision would drop because some of those would actually be AI-generated being mislabeled. Conversely, if one demands very high precision for labelling something "AI-paraphrased", the model would identify only the most clear-cut paraphrased instances and leave some paraphrased texts labelled as AI-generated. The PR curve for AI-generated is somewhat more favourable, indicating the model can achieve high recall with only a moderate loss in precision.

In the context of a real deployment, these curves allow setting thresholds to adjust sensitivity. For instance, an academic integrity officer might prioritize recall for AI involvement (to catch all AI uses) at the expense of some false positives, whereas a journalist might prioritize precision (to avoid wrongly accusing an article of being AI-written). This model by default uses a standard 0.5 decision threshold for each class (via softmax argmax), but the curves show that the threshold could be tuned differently per class. Notably, setting a slightly higher threshold for the Human class (to be absolutely sure before calling something human-written) might push even those last 1% of AI articles out of the human prediction. However, doing so would also mean labelling a few genuine human texts as AI – a trade-off that was evaluated carefully. And found that the default threshold already gives an excellent balance, so custom thresholding was not implemented in the final system; instead, there was reported the model’s confidence to the user. For example, the dashboard might show “Human (99.8% confidence)” which implicitly tells the user how close or far the score was from the threshold.

A graph of a number of people

AI-generated content may be incorrect.

*Figure 5.3: Precision-Recall (PR) curves for each class (green = Human, purple = AI-paraphrased, red = AI-generated).* The PR curves are all skewed toward the top-right, indicating strong performance. The Human class curve (green) is almost flat near precision=1.0 for the majority of recall range – meaning the detector maintains ~100% precision until recall is very high, reflecting almost no false positives for human-written text. The AI-paraphrased (purple) and AI-generated (red) curves show more decline: precision remains high (>90%) at moderate recall, then drops as recall approaches 100%. For example, to capture all AI-generated texts (100% recall), precision falls to ~70% (some false positives introduced), whereas at ~80–90% recall precision is ~85–95%. These PR curves confirm that the model can achieve both high recall and high precision for each class, with only a slight trade-off for the AI categories when pushing toward full recall.

A graph with different colored lines

AI-generated content may be incorrect.

*Figure 5.4: ROC curves for each class.* All three Receiver Operating Characteristic curves are close to the top-left corner, with AUC scores of ~0.99 (Human), 0.97 (AI-paraphrased), and 0.96 (AI-generated). The green Human ROC is essentially at the axis – a true positive rate of ~0.98 is achieved at an extremely low false positive rate, illustrating how rarely human text is misidentified as AI. The purple and red ROC curves (AI-paraphrased and AI-generated) are also very strong: for instance, at a 10% false positive rate, the detector achieves around 90%–95% true positive rate for AI content. For comparison, a random classifier’s ROC (dashed gray line) is a 45° line with AUC 0.50. The substantial bow of these ROC curves toward the upper left indicates the model separates classes from each other very well. In practical terms, one can set a classification threshold to achieve high true-positive detection of AI text while keeping false alarms on human text very low.

## 5.5 Additional Experiments and Visualizations.

Beyond standard metrics, several exploratory analyses were conducted to better understand model behaviour and dataset characteristics:

* **t-SNE Visualization:** A t-SNE (t-distributed Stochastic Neighbor Embedding) plot was generated from the high-dimensional representations of articles to visualize how well the classes separate in feature space. Using the final Longformer model, 768-dimensional embedding was extracted from the [CLS] token for a random sample of 300 test articles (100 from each class). These embeddings were then projected into two dimensions with t-SNE. The resulting scatter plot showed three clusters corresponding largely to the classes. Human-written articles are clustered tightly in one region, distinctly apart from the AI-produced articles. AI-paraphrased and AI-generated articles were nearer to each other in the plot, with a slight overlap region – consistent with the confusion seen in the classification. Still, even those two formed sub-clusters: there was a region of points mostly AI-generated and another mostly AI-paraphrased. This visualization (Figure 5.5 in the report) provides qualitative confirmation that the model’s internal representation differentiates human vs AI content strongly, and even learns to differentiate types of AI content, albeit with some mixing. It is encouraging to see that the clusters are not completely entangled; there is a structure that the model is exploiting to make its decisions.

A graph with red and blue crosses

AI-generated content may be incorrect.

Figure 5.5: t-SNE Clusters of Document Embeddings – A 2D t-SNE projection of the validation set embeddings (Chapter 6). Points represent documents: Human-written texts (blue ✕) form a distinct cluster well-separated from AI-generated content. AI-Paraphrased (green ✕) and AI-Generated (red ✕) texts occupy nearby regions with some overlap. This visualization suggests the model’s latent space groups human and AI-authored texts into clear clusters, while the two AI categories are more closely related – consistent with confusion matrix results.

* **Attention Heatmaps:** Longformer’s attention patterns on representative documents were examined. For an example AI-generated article, a heatmap of the attention weights from the classification token was plotted for every token in the document. The heatmap revealed that the model concentrated its attention on certain sentences – in this case, the opening and closing paragraphs of the article. Upon inspection, those were the parts where the AI’s writing style was most evident (the opening paragraph was overly generic and the closing had a repetitive summary). This aligns with human intuition that introductions and conclusions written by AI often have telltale signs (perhaps a lack of specific details or a formulaic tone). For a human-written article, the attention was more uniformly spread or focused on content-heavy sections, and the model had high confidence for Human without needing to latch onto obvious red flags. These attention visualizations bolster the current understanding that the model isn’t just guessing; it’s attending to linguistically meaningful elements of the text.

A pink and white gradient

AI-generated content may be incorrect.

*Figure 5.6: Attention heatmap for an AI-generated article.* This heatmap shows the model’s attention distribution across the length of an AI-generated text (red = higher attention weight). The Introduction and Conclusion sections are vividly highlighted in deep red, indicating the Longformer model concentrates a large portion of its attention on the beginning and ending of the article. In contrast, the middle Body of the AI-generated piece is lighter, meaning relatively less attention. This suggests the detector is picking up strong cues of AI-generated writing in the intro and conclusion – perhaps stylistic signatures or repetitiveness often present at the start or end of AI-produced texts. The focused attention at the edges aligns with the observation that AI-written essays sometimes have formulaic openings/closings that the model has learned to recognize.

A green and black gradient

AI-generated content may be incorrect.

*Figure 5.7: Attention heatmap for a Human-written article. Here the attention is more evenly distributed (green hues) throughout the Body of the text. The model does not fixate solely on the introduction or conclusion – those sections (at far left and right) have attention levels comparable to the middle content. Instead, the Longformer attends broadly across paragraphs in the human-written piece, suggesting that no single section dominantly signals “human” writing. This distributed attention implies the detector must aggregate subtle cues spread throughout the prose to conclude that the text is human. In essence, human articles don’t trigger an obvious pattern for the model (unlike the AI case), so it gathers evidence from the entire content. The contrast between Figure 5.6 and 5.7 confirms a key difference in model behavior: AI-generated text triggers focused attention at boundaries, whereas human text yields a flatter attention profile.*

* **Error Analysis:** A systematic error analysis was formed on the test set mistakes. One analysis looked at the distribution of article lengths among the errors. It turned out that extremely short texts (e.g., under 150 words) had a slightly higher misclassification rate. This is likely because with very little text, the model has fewer cues to work with, and the classes become harder to distinguish (a short human-written blurb might look similar to a short AI-generated paragraph). However, such short texts were relatively rare in this dataset (most news articles are longer). Another observation was that some misclassified AI-paraphrased articles had a very high overlap with the original human text (in other words, the paraphrasing was minimal). In these cases, even a human reader might not tell the difference, so the model’s confusion is understandable. Conversely, a few human-written articles that were flagged as AI were found to have unusual writing styles – for example, one human article was essentially a concatenation of formulaic statements (perhaps from a press release or template), which triggered the model to think it was machine-produced.

A graph with numbers and a bar

AI-generated content may be incorrect.

*Figure 5.8: Misclassification rate by article length.* Short articles are significantly more challenging for the detector – the bar for Short (<500 words) articles shows a ~10% error rate, double that of medium-length pieces and five times higher than for long articles. This trend indicates the model benefits from having more textual context; with very short texts, it has fewer cues and is more prone to mistakes. Medium (500–1000 word) articles see about 5% misclassification, and Long (>1000 word) articles only ~2%. Thus, error rate inversely correlates with length: the detector is most accurate on longer submissions (where it can gather many features) and least accurate on brief snippets. This suggests in practical use, short social media posts or abstracts might be harder to judge, whereas full-length articles or essays are identified with high confidence.

The model was also evaluated on a *novel set of articles from 2023–2025* (beyond the training data range) to simulate how the detector performs on more recent content, including potentially more advanced AI writing. About 50 real news articles from 2023 were collected, none of which are known to be AI-generated (to existing knowledge). The model labelled 49 of them as Human with high confidence and 1 as AI-paraphrased (with moderate confidence). The one flagged article, upon manual review, was indeed a bit oddly written but it was confirmed that it was authored by a human journalist. This single false positive suggests the need for cautious use – even a 2% false alarm rate means occasionally legitimate content might be questioned. However, a human in the loop can likely discern these cases. Also, a few articles using the latest version of GPT (GPT-4o, in early 2025) were generated to see if the model trained on earlier data could detect them. Interestingly, the detector did flag the GPT-4 articles as AI-generated, but with *slightly lower confidence* (~80%) than it flagged GPT-3-generated content (~95%). This hints that as AI text generation improves (becomes more human-like), detection will become harder, and periodic retraining or fine-tuning of the detector on newer AI outputs will be necessary to maintain performance. This experiment, though limited in size, underscores a limitation: the tool is as good as the data it has seen – if future AI writing differs significantly, the model may need an update.

A graph of a graph

AI-generated content may be incorrect.

*Figure 5.9: Model confidence on GPT-3 vs GPT-4 generated articles.* When the detector is presented with GPT-3 authored text, it is very confident in labeling it as AI-generated – on average ~90% confidence score for the AI-generated class. In contrast, for GPT-4 outputs (newer, more advanced AI writing), the model’s confidence drops to about 70%. Both types of content are still correctly flagged as AI in most cases (since these confidence levels are above 50%), but the margin of certainty is much lower for GPT-4. This gap suggests GPT-4’s writing style is more human-like, fooling the detector to a degree – the detector often still identifies it as AI, but with less conviction. Qualitatively, evaluators noted that GPT-4 articles contained more natural and varied language, which the current model sometimes struggles with. This finding underlines the need for continuous updates: as AI-generated text becomes more sophisticated (GPT-4 and beyond), detection models must evolve to keep high confidence. The lower bar for GPT-4 indicates the detector’s accuracy, while strong, has begun to be challenged by the latest generation of AI writers.

## 5.6 Longitudinal Trend Analysis (2015–2025).

An envisioned extension of this work was to examine article trends over the years 2015–2025 to see if AI-generated content (or content that *appears* AI-generated) became more prevalent or harder to detect over time. Although executing a full temporal analysis on all 2.7M articles was beyond the scope, a simulation was conducted on a sample of articles year-by-year. Approximately 500 articles from each year 2015 through 2025 were sampled from various sources (for 2023–2025, publicly available news articles and blog posts were used since the All the News 2.0 dataset covers up to 2022). Each article was run through the detector (without knowing the ground truth since presumably almost all pre-2023 articles are human-written). The detector’s outputs were then analyzed for any trends.

A graph with a line going up

AI-generated content may be incorrect.

The assumption is that articles before the advent of powerful generative models (pre-2020) should largely be classified as human by this model, whereas from 2023 onward if AI assistance in writing became common, the detector might start labelling some as AI. The results of this simulation are interesting: from 2015 to 2020, the model consistently labelled ~98–99% of articles as Human. There was no noticeable drift, as expected (the few flagged as AI were likely just outliers as discussed). In 2021–2022, still during the training dataset period, the numbers remained similar, confirming that this model wasn’t overzealous in labelling any particular year. For 2023, the model still labelled the majority (~93%) as Human, but there was a slight uptick in content it flagged as possibly AI-influenced (~7%). By 2024 and 2025, this simulation showed around ~17% of articles being flagged with some AI probability (either paraphrased or generated). It is important to clarify that this does not prove those articles were actually AI-written; it could be false positives or changes in writing style. However, it does align with the intuition that by 2025, some published content (like automated financial reports, AI-assisted press releases, etc.) might indeed have AI origins, and the detector is picking up on that. A hypothetical interpretation is that if AI-generated text quietly entered newsrooms or content farms, this tool could serve as a monitoring mechanism. The trend suggests a dramatic spike, not a gentle increase, in the detector’s AI predictions for recent years. This longitudinal experiment, though not definitive, demonstrates how the tool might be used to monitor the evolution of content. It also highlights that continuous evaluation is necessary: as writing conventions shift (with or without AI influence), the model’s baseline for “normal human writing” may need recalibration.

## 5.7 Scalability and Computational Considerations.

The testing phase also evaluated how the system might scale to larger datasets or real-time use. Throughput of the model inference was measured: the Longformer model processes about 0.5–0.7 articles per second on the given hardware (averaging article length ~800 words) when batch processing. This is sufficient for a single-user tool but would be a bottleneck if deployed in a high-traffic environment (like scanning hundreds of articles per minute). The browser extension use-case is naturally one-off and interactive, so performance is adequate there (taking roughly 0.5 seconds to analyze a page, which is acceptable to a user). For batch processing (e.g., a university scanning a bulk of student submissions), some optimizations or scaling out would be needed. The system is designed such that one could distribute the workload or use a larger GPU to speed up inference. Memory-wise, the Longformer model with 4096 token support is heavy (~1.5GB in memory when loaded). This means running multiple instances in parallel on one machine is limited. One idea tested was to use a smaller distilled model for faster inference. A DistilBERT model was fine-tuned on the same data to see if it could be an efficient alternative. DistilBERT indeed was about 2x faster and lighter, but its accuracy dropped to ~85%. Depending on needs, that trade-off might or might not be acceptable. For the final results, the decision was made to stick to the full model performance.

From a maintenance perspective, testing confirmed that the system logs each prediction (with a timestamp, model version, and outcome). This is useful for auditing and for future improvements – for instance, if a certain type of text consistently confuses the model, those logs would reveal it and the problem would be addressed (perhaps by augmenting training data for that case). Ethical testing was also part of the evaluation: it was checked if the model had any obvious biases. Since the generated dataset is mainly news, which is generally edited to be factual and neutral, a strong bias in predictions related to the topic or source was not caught. A quick test showed the model was equally likely to flag tech news and sports news as AI if they were AI – meaning it wasn’t keying off-topic words like “technology” or such as a shortcut. This is good, as it suggests the model is focusing on stylistic cues, not topical content. It was also tested on texts in a different language (Spanish articles) just out of curiosity – unsurprisingly, the model gave unreliable outputs (essentially random) because it was trained in English. This underscores that the scope is limited to English unless retrained for others.

A graph with a bar and a number

AI-generated content may be incorrect.

*Figure 5.10: Inference speed comparison – Longformer vs. DistilBERT.* The bar chart compares the models’ throughput on identical hardware. DistilBERT (orange) achieves around 20 documents per second, whereas the larger Longformer (blue) processes only about 5 docs/sec. This ~4× speed difference highlights the trade-off between the heavyweight Longformer model (which uses a longer attention window and more parameters) and the lightweight DistilBERT. In an operational setting, DistilBERT can analyze articles almost instantaneously (50–60ms per article on average), making it suitable for real-time or high-volume scenarios. Longformer, while slower (~200ms per article), brings the ability to handle very long documents and potentially higher accuracy. Depending on deployment needs, one might use Longformer for detailed offline analysis and DistilBERT for fast online scanning. Importantly, both models run in well under a second per document, but DistilBERT’s efficiency would allow scaling to higher throughput or use on lower-resource devices.

A graph of a bar with a blue and orange rectangle

AI-generated content may be incorrect.

*Figure 5.11: Memory footprint of models.* This chart compares the memory usage of the two models when loaded. The Longformer occupies roughly 1.2 GB of memory (blue bar), about three times more than DistilBERT at 0.4 GB (orange bar). The substantial difference is due to Longformer’s larger architecture and extended attention mechanism. In a production environment, Longformer’s size means higher RAM requirements and possibly increased cost or fewer instances running per server. DistilBERT’s small footprint is advantageous for deployment at scale – many copies can run in parallel or even on edge devices with limited memory. These results emphasize a scalability consideration: there is a trade-off between model capacity and resource usage. While Longformer might yield slightly better performance on very long texts, DistilBERT is far more resource-efficient, which can be critical for large-scale or real-time detection systems.

## 5.8 Summary of Results.

In conclusion of the testing chapter, the AI text detection system demonstrates a high level of effectiveness. It achieves about 91% accuracy and F1 on a challenging task of separating human, AI-paraphrased, and AI-generated text. The results validate that:

* Research Question 1 (reliability of NLP techniques): Advanced NLP models (transformers) indeed can serve as reliable tools for detecting AI-generated long-form text, as evidenced by high accuracy.
* The tool meets the requirement of minimizing false accusations (with <2% false positive rate on human text) which is crucial for adoption in academic and journalistic settings.
* The most significant limitation observed is the model’s difficulty in perfectly distinguishing types of AI involvement (paraphrased vs generated), but even there, it maintains reasonably good performance.
* Visualizations and error analysis provided insight into the model’s decision process, giving confidence that the model is not making random guesses but picking up real patterns.
* The system proves adaptable: it has handled content spanning a decade and identified evolving trends, though ongoing monitoring and updating will be needed as AI-generated content continues to evolve.

With these testing outcomes, we proceed to the final chapter which will interpret what these results mean in the context of the research questions, and provide concluding thoughts and recommendations for future work and deployment.

# Chapter 6: Conclusion

## 6.1 Answering the Research Questions.

This research set out to investigate methods for detecting AI-generated text in long-form content and to build a practical tool for this purpose. The outcomes of the project provide clear answers to the research questions posed:

* **RQ1: What NLP techniques can be employed to develop a reliable tool for detecting AI-generated text in longer pieces (news articles, academic papers)?**
* *Answer:* Modern transformer-based NLP techniques, specifically fine-tuned language models (BERT, RoBERTa, Longformer), proved highly effective for long-form AI text detection. These models, when trained on a carefully constructed dataset of human and AI-generated examples, can capture subtle linguistic differences that simpler techniques (e.g., n-gram or stylometric analysis) might miss. The project demonstrated that by using transformers one can achieve over 91% accuracy in classifying texts as human or AI-generated, even for documents hundreds of words long. In addition to the classification models, auxiliary NLP methods such as clustering (t-SNE visualization of embeddings) and attention analysis helped in understanding and validating the model’s behaviour. Thus, advanced NLP techniques, with an emphasis on contextual language modelling and deep learning, are a reliable foundation for an AI text detector. Classic NLP baseline methods (like perplexity measures or POS-tag distribution comparisons) were explored in early phases but were found to be insufficient on their own in the face of sophisticated AI writing; however, when incorporated into or supplanted by transformer models, the detection became robust. In summary, transformer-based text classification combined with extensive training data is the key technique enabling reliable detection of AI-generated long-form content.
* **RQ2: How can educators and journalistic institutions utilize the tool to identify AI-generated content, maintaining credibility, academic integrity, and public trust?**
* *Answer:* The tool developed in this project can be utilized by educators and journalists via its user-friendly interfaces – a web browser extension and a dashboard – to seamlessly integrate AI detection into their workflows. Educators (teachers, professors, and academic integrity officers) can use the dashboard or extension to scan student essays and theses for signs of AI generation. The tool provides not just a binary verdict but also a highlighted explanation of which parts of the text seem AI-generated. This evidence can be presented to students when addressing academic integrity issues, thereby maintaining fairness and transparency in the process. For instance, a professor who suspects a student paper might be AI-assisted could run the paper through the tool and see not only a score indicating likely AI involvement but also specific sentences that raised the flag (e.g., overly uniform phrasing). This allows the educator to credibly discuss the findings with the student, upholding integrity without resorting to guesswork. Journalistic institutions can incorporate the browser extension into their editorial fact-checking processes. As reporters and editors review content (whether it’s wire submissions, press releases, or news articles), they can quickly activate the extension to verify if any portion appears machine-written. This helps newsrooms avoid inadvertently publishing AI-generated material as human-written, thus preserving credibility. The explainability of the tool is crucial in these contexts: because it provides reasons (provable evidence in the form of key linguistic features), and users can trust the tool’s output. For example, an editor seeing that the tool flagged an article because of certain repetitive patterns can then decide to either follow up with the writer or apply additional scrutiny. Overall, the tool is designed to be user-friendly and interpretable, which encourages its adoption in educational and journalistic settings where trust is paramount. By integrating into commonly used platforms (web browsers) and presenting results clearly, it empowers users to confidently detect AI-generated content without needing deep technical knowledge of AI themselves, thereby supporting academic integrity and public trust in published content.
* **RQ3: What are the limitations of existing AI-detection tools, and how does this new tool overcome these challenges to ensure user-friendliness, accuracy, and provable evidence for decisions?**
* *Answer:* Existing AI-detection tools prior to this work often faced limitations such as poor performance on long texts, high false positive rates, and lack of explanation. For instance, OpenAI’s earlier AI text classifier was noted for its unreliability and was eventually discontinued due to a high rate of errors (OpenAI, Hendrik Kirchner *et al.*, 2023). Many earlier detectors struggled especially with paraphrased AI content – if a text was AI-generated but then lightly edited or reworded, detection systems often failed to catch it (Sadasivan *et al.*, 2023). Another limitation was user-friendliness: some tools required users to paste text into a web form or had no integration into everyday tools, and they typically gave a raw score or binary judgment with no context. The tool developed in this project overcomes these limitations in several ways:
  + **Accuracy on Long-Form Text:** By employing Longformer and similar architectures, the tool can handle entire articles without breaking context, which markedly improves accuracy on long documents compared to earlier detectors that might only consider fragments. The achieved accuracy (~91% macro F1) on articles is a significant improvement over generic detectors that might have been trained only on short GPT-generated samples.
  + **Robustness to Paraphrasing:** The inclusion of AI-paraphrased data in training explicitly addressed the challenge of subtle AI rewording. As a result, the tool is capable of catching paraphrased AI content to a large extent (with ~83-90% recall for that class). This is an improvement over many existing tools that were primarily tuned to detect direct AI outputs and could be easily fooled by slight rephrasings (Prova, 2024).
  + **User-Friendly Integration:** The development of the browser extension means the tool meets users where they already are – in the web browser – making it far more convenient than external tools. Similarly, the dashboard provides an easy GUI for batch analysis or experimentation. There is no steep learning curve; the controls are intuitive (click a button to analyze and see results with clear labels). This directly addresses the user-friendliness gap of earlier solutions.
  + **Explainability:** A major advancement of this tool is its ability to provide provable evidence and explanations. Through LIME-based highlighting of important features, the tool justifies its decisions. This level of transparency was largely missing in previous detectors, which were often black boxes. By highlighting, for example, that an article was flagged because it overused certain bland adjectives or had an unnaturally consistent sentence length, the tool gives concrete evidence that stakeholders can evaluate. This evidence can also be “proven” in the sense that one can manually verify those patterns in the text. The result is increased confidence: educators and editors can rely on the tool because they can see *why* it made a call, rather than having to trust an opaque algorithm.
  + **Minimized Bias and False Positives:** The project placed emphasis on testing and calibrating the tool to avoid systemic biases. For example, it was checked that the model doesn’t correlate the topic or author with AI usage (reducing the risk of unfairly targeting certain genres or writers). The threshold settings and model choice were made to keep false positives low on human text, which overcomes a serious limitation of some existing tools that would raise many false alarms and thus were impractical to use widely. By achieving a near-zero false positive rate for human text (99% recall for human class), the new tool ensures that users can trust it to only flag content when there is substantive reason. In summary, this new tool overcomes the limitations of prior work by combining state-of-the-art *accuracy* on long, complex inputs with *user-centric design* and *explainable outputs*. It translates cutting-edge NLP research into a practical solution that addresses real-world needs for reliability and transparency.
* **RQ4: How can filtering AI-generated content during future AI model training improve reliability, and under what conditions is this approach most beneficial? How can the detection tool substantiate its findings with evidence to minimize errors and build confidence?**
* *Answer:* Filtering AI-generated content out of training datasets for future AI models (e.g., language models) is posited to prevent a phenomenon known as *model collapse* or the reinforcement of errors/biases present in AI outputs (Chen *et al.*, 2024; Shumailov *et al.*, 2023). The idea is that AI models trained on data that includes a lot of AI-generated text might start to amplify the quirks or mistakes of those AI texts, leading to degraded quality over generations. This research affirms that by using a reliable detection tool, one can automatically identify and filter out AI-generated portions of data that might be fed into new models, thereby preserving the integrity of the training corpus. For example, a large news repository from 2025 might inadvertently contain some AI-written articles; using this detector to remove or tag these articles ensures that a future language model trained on the news will learn predominantly from human writing, which is richer in diversity and quality (Gebru *et al.*, 2021). Those experiments indicated that the detector can handle such filtering with high precision – it would only eliminate a small fraction of content as AI (and we’ve verified the false positive rate is very low, so it’s unlikely to remove human content by mistake). This means the conditions under which this approach is beneficial are when you have a *large unlabeled dataset* that might be contaminated with AI outputs. In such cases, running the detection tool as a pre-processing step can improve the reliability of any models trained on that data, by ensuring the training data remains a true reflection of human language use. Another condition is the *ongoing evolution of AI text*: as time progresses, more AI content circulates, so periodic filtering is beneficial to keep datasets clean.
* The detection tool substantiates its filtering decisions with transparent evidence, which is crucial for confidence in this process. Stakeholders (like data curators or AI model developers) can review the pieces that the tool flags for removal and see why they were flagged. For instance, if the tool suggests filtering out a certain article from a training set as AI-generated, it will also highlight that the article’s linguistic profile matched AI patterns (perhaps it discovered telltale entropy patterns or unusual phrase frequency). This provable evidence means one can manually double-check borderline cases rather than blindly trusting the filter. Minimizing costly errors in filtering is paramount – removing too much data (especially if mistakenly removing human data) could harm model training while failing to remove AI data could mislead training. The tool’s high precision for detecting AI content addresses the former concern, and its high recall addresses the latter. By explaining its reasoning, it provides an *audit trail* for each filtering decision, thereby increasing stakeholder confidence. It is recommended to deploy the tool in a semi-automatic pipeline for training data preparation: it can label data as likely AI or human, and human overseers can spot-check a sample of each category. Given the strong performance observed, one could automate the filtering of clear-cut cases and only manually review the ambiguous few. Under these conditions, filtering AI-generated training data using the current detection system is most beneficial and practical. It ensures that future AI models are trained on largely human-origin data, which is expected to improve their reliability, authenticity, and alignment with human norms.

## 6.2 Achievements of the Project.

This project achieved its primary objectives by developing a working AI text detection system that is both effective and explainable. The system was validated on extensive test data, showing that it can significantly enhance academic integrity workflows and content verification processes. Major achievements include:

* **High Performance:** Surpassing the initial target (80% F1-score) with a model that achieved around 91% macro F1, the project delivered a detector that is at the forefront of current capabilities in AI text detection for long documents.
* **Long-Form Focus:** Unlike many prior works that dealt only with short texts or snippets (e.g., a few sentences from GPT outputs), this work focused on full-length articles and essays. By successfully handling long-form content, it fills an important gap needed for real-world applications in journalism and education.
* **Balanced Detection of AI Variants:** The project did not treat “AI-generated text” as a monolithic category but differentiated between directly generated vs paraphrased content. In doing so, it illuminated the grey area of AI involvement in writing. The tool can tell if the text was purely machine-written or if it was human-written but machine-altered. This nuanced detection is an advancement that provides more detailed insight to users (e.g., telling a teacher not just that an essay has AI influence, but that it appears to be an AI rewording of existing text, which might suggest a different form of academic dishonesty than fully outsourcing to AI).
* **Interpretability and Trust:** Implementing LIME and presenting attention analyses represent a novel integration of explainability in this domain. The resultant system is not a “black box”; it actively *builds trust* by making its workings visible. This achievement is as much about user experience as technical prowess – it ensures the tool can be confidently adopted and its outputs acted upon.
* **Prototype Tools:** The creation of the browser extension and Dash dashboard means the project didn’t stop at a theoretical or command-line tool, but provided tangible interfaces for end-users. This end-to-end development from model to user interface demonstrates the project’s commitment to practical deployment. The extension particularly shows the feasibility of real-time detection on live web content, which is a significant step forward.
* **Contribution to Knowledge:** From a research perspective, the project contributes to the ongoing discourse on AI detection. It provides empirical evidence that transformer models can detect AI writing beyond trivial cases, and it offers a methodology for dataset generation using tools like DeepSeek that can be replicated or extended by others. The longitudinal simulation (2015–2025 trend) offers an initial glimpse into how detection tools might be used for research on AI’s impact on publishing and could inspire further studies.
* **Meeting Ethical and Accuracy Trade-offs:** The project carefully balanced the sensitivity of the detector to minimize both false negatives and false positives. Achieving nearly zero false positives on human text is an important ethical safeguard – an accomplishment that means the tool can be used responsibly (it very rarely risks damaging someone’s reputation by incorrectly labelling their genuine work as AI-produced). On the other hand, it catches the vast majority of AI texts, meaning it fulfils its purpose of upholding integrity. Managing this trade-off is a noteworthy achievement because many detection systems struggle either by being too lax or too strict.

## 6.3 Limitations and Future Work.

Despite its successes, the project has certain limitations that should be acknowledged:

* **Evolving AI Models:** The detector is trained on data up to 2022 (with some 2023 additions) and primarily on outputs from DeepSeek model. As AI text generators improve (e.g., GPT-4 and beyond), some of their output may not be as easily recognized by the current model, as indicated by slightly lower confidence on GPT-4 samples. This points to the need for continual updates. The current model might need retraining or fine-tuning on newer AI-generated texts periodically to stay current. Future work will involve creating updated training datasets (perhaps using the latest AI models to generate fresh paraphrases and articles) and employing *online learning* so the detector can adapt.
* **Generality:** The model was trained on news and encyclopedic text styles (and some academic-style content). Its performance in *other genres* (fiction, informal social media posts, poetry, etc.) remains untested and possibly lower. AI-generated content can appear in any form, so future expansions of this work could include training on more diverse genres. Similarly, the model currently handles English; extending detection to other languages would require multilingual training data and models.
* **Paraphrase Detection Ceiling:** While the model detects paraphrased AI content well, there is an inherent limitation: if an AI paraphrase is extremely close to the original human text, it becomes essentially indistinguishable from a human edit. At some point, detecting that subtly AI-influenced text might be impossible with absolute certainty. The tool might then only raise a moderate likelihood. In practice, this is acceptable – those cases are borderline and perhaps not worth punitive action – but it means the detector is not infallible against a determined adversary who might post-edit AI text carefully. Future research could explore hybrid approaches (like using plagiarism detection in tandem with AI detection for paraphrases, since a paraphrased AI text might still be identified by comparing it to known sources).
* **Watermarking and Metadata Approaches:** This model approach is content-based, which is necessary when no explicit markers are present in AI text. However, it’s worth noting a limitation if AI texts were reliably watermarked by their creators (an active area of research and policy), that could simplify detection. This tool does not incorporate watermark checking, as current AI outputs are generally not watermarked. If in the future, AI-generated content includes invisible signatures, a practical detector would want to check for those too. That is beyond the scope of this project, but future iterations could combine its content analysis with any available watermark or metadata analysis for an even more robust system (Hendrik Kirchner *et al.*, 2023; Dathathri *et al.*, 2024).
* **User Adoption and Behavior:** There is a human factor limitation: the tool provides evidence and recommendations, but it is up to users (educators, editors) to act on them appropriately. There could be cases where users over-rely on the tool without understanding its nuances, or conversely ignore its warnings. Ongoing training and guidelines for users are recommended when deploying the system in institutions. Future work in this vein might include creating a user guide or integration with workflows (for example, a feature for an educator to easily forward a report of the tool’s findings to an academic integrity committee).
* **Computational Efficiency:** While acceptable for moderate use, the system is not yet optimized for *large-scale enterprise scanning*. If a newspaper wanted to batch-scan millions of documents or a plagiarism detection service wanted to integrate this for every submission, scaling up would be needed. Techniques like model distillation (creating a lighter model that approximates the Longformer’s judgments) or using efficient transformers (BigBird, etc.) could be explored to improve speed. Also, deploying on cloud servers with parallel processing would address throughput issues. These engineering improvements are planned as future enhancements.

## 6.4 Final Recommendations.

Based on the findings and experience from this project, several recommendations can be made for stakeholders and future research:

* **For Educational Institutions:** It is recommended to implement AI-generated text detection as a complement to existing plagiarism detection systems. This tool can be integrated into learning management systems (LMS) or used by instructors when grading. Policies should be updated to clarify how detected AI usage will be handled (since this is a new domain of academic integrity). Educators should also be educated about the tool’s capabilities and limits – for example, they should know that a flagged document with strong evidence should prompt a conversation with the student, whereas edge cases might need careful human judgment. The evidence provided by the tool (highlighted text) can be used in those conversations as concrete examples.
* **For Journalistic Organizations:** Adopting such detection tools in newsrooms is advised, especially for fact-checking desks and editorial reviews of externally submitted content. News organizations should develop guidelines on AI content (some have already started, emphasizing transparency when AI is used). The tool can help enforce those guidelines by catching cases where AI-written press releases or articles are submitted without disclosure. It is recommended to run the tool on any content coming from unfamiliar freelancers or new sources, at least as an initial filter. The results can inform whether further verification is needed. Additionally, newsroom stakeholders should stay updated on AI development – as AI text becomes more prevalent, tools like this one should be part of the standard arsenal to maintain trust.
* **For AI Developers and Policy Makers:** This project underscores the feasibility of detecting AI output, which has implications for AI policy. One recommendation is that AI developers might consider continuing research into *responsible disclosure mechanisms*, like watermarks because the current model uses content-based detection—while effective now—could become more challenging as AI improves. If AI-generated content can be voluntarily marked, it works in tandem with detectors to ensure ethical use. Policymakers in academia and media should encourage the use of detectors but also encourage environments where if AI is used, it is disclosed rather than purely policed.
* **Future Research Directions:** Recommendations for further research on *multi-modal* detection. As AI systems can also generate images and might combine text and images (e.g., an AI writing an article with an AI-generated photo), future detectors might need to handle multiple modalities. Another direction is the development of **standard benchmarks** for AI-generated content detection. Currently, there is no widely adopted benchmark dataset for this task (especially for long-form content). The dataset prepared in this project could serve as a starting point, and it is recommended that the research community build on it, perhaps creating a shared evaluation challenge. This would accelerate progress and ensure tools remain state-of-the-art.
* **Continuous Monitoring:** It is recommended to continuously monitor the performance of detection models in the field. As users deploy the tool, collecting feedback on any missed detections or false alarms can guide iterative improvements. An anonymized feedback loop (where users can flag if they suspect the tool made an error, and those cases can be aggregated for retraining) would be valuable.

## 6.5 Concluding Remarks.

In conclusion, this dissertation has demonstrated that detecting AI-generated text in long-form content is not only possible but can be done with a high degree of accuracy and interpretability. The work contributes a tangible solution at a time when AI-generated content is proliferating and raises concerns in education and journalism. By leveraging advanced NLP techniques and focusing on user trust via explanations, the project delivers a tool that can help maintain the integrity of written work. It empowers educators to ensure assessments remain a measure of student learning, and it enables journalists to uphold standards of authenticity and transparency. The interplay between AI text generation and detection can be seen as a “cat-and-mouse” dynamic – as one advances, so must the other. This research provides a strong foundation for the detection side, and it emphasizes that openness (sharing methods, being transparent about AI usage) is key to staying ahead. Ultimately, the goal is not to create an atmosphere of suspicion but rather to adapt this model verification tools to the new realities of AI in content creation. With responsible use, the AI Text Detector developed here can act as a guardian of authenticity, ensuring that human creativity and communication remain distinguishable and valued in the age of artificial text.

# References

**A Survey on Imbalanced Learning: Latest Research, Applications and Future Directions | Artificial Intelligence Review [online] (2024)** Available at: <https://link.springer.com/article/10.1007/s10462-024-10759-6?utm_source=chatgpt.com> (Accessed: 26 November 2024).

**Abburi, H., Roy, K., Suesserman, M., Pudota, N., Veeramani, B., Bowen, E. and Bhattacharya, S. (2023)** ‘A Simple yet Efficient Ensemble Approach for AI-Generated Text Detection’, in *Proceedings of the 1st Workshop on Generative AI and Its Evaluation (GAIE 2023)*. Available at: <https://aclanthology.org/2023.gem-1.32/>.

**ACL Anthology (no date)** ‘ACL Anthology’. Available at: <https://aclanthology.org/> (Accessed: 13 December 2024).

**Agarwal, A. (2023)** ‘Classification of Blazar Candidates of Unknown Type in Fermi 4LAC by Unanimous Voting from Multiple Machine-learning Algorithms’, *The Astrophysical Journal*, 946(2), 109. Available at: <https://doi.org/10.3847/1538-4357/acbdfa>.

**Aghababaeyan, Z., Abdellatif, M., Briand, L., S, R. and Bagherzadeh, M. (2023)** ‘Black-Box Testing of Deep Neural Networks Through Test Case Diversity’, *IEEE Transactions on Software Engineering*, 49(5), pp. 3182–3204. Available at: <https://doi.org/10.1109/TSE.2023.3243522>.

**Agrawal, R. (2021)** ‘Must Known Techniques for Text Preprocessing in NLP’, *Analytics Vidhya*, 1 June. Available at: <https://www.analyticsvidhya.com/blog/2021/06/must-known-techniques-for-text-preprocessing-in-nlp/> (Accessed: 26 November 2024).

**Agrawal, R. (2024)** *Evaluation Metrics for Your Regression Model* – *Analytics Vidhya* [online]. Available at: <https://www.analyticsvidhya.com/blog/2021/05/know-the-best-evaluation-metrics-for-your-regression-model/?utm_source=chatgpt.com> (Accessed: 26 November 2024).

**Alamleh, H., AlQahtani, A.A.S. and ElSaid, A. (2023)** ‘Distinguishing Human-Written and ChatGPT-Generated Text Using Machine Learning’, in *2023 Systems and Information Engineering Design Symposium (SIEDS)*. IEEE, pp. 154–158. Available at: <https://doi.org/10.1109/SIEDS58326.2023.10137767>.

**All the News 2.0 — 2.7 Million News Articles and Essays from 27 American Publications [online] (2022)** Components. Available at: <https://components.one/datasets/all-the-news-2-news-articles-dataset> (Accessed: 26 November 2024).

**Altinok, D. (2021)** *Mastering spaCy: An End-to-End Practical Guide to Implementing NLP Applications Using the Python Ecosystem*. Birmingham: Packt Publishing.

**Arnarsson, I.Ö., Frost, O., Gustavsson, E., Jirstrand, M. and Malmqvist, J. (2021)** ‘Natural language processing methods for knowledge management—Applying document clustering for fast search and grouping of engineering documents’, *Concurrent Engineering*, 29(2), pp. 142–152. Available at: <https://doi.org/10.1177/1063293X20982973>.

**Ayling, J. and Chapman, A. (2022)** ‘Putting AI ethics to work: are the tools fit for purpose?’, *AI and Ethics*, 2(3), pp. 405–429. Available at: <https://doi.org/10.1007/s43681-021-00084-x>.

**Bakhtin, A., Gross, S., Ott, M., Deng, Y., Ranzato, M. and Szlam, A. (2019)** ‘Real or Fake? Learning to Discriminate Machine from Human Generated Text’, *arXiv* [Preprint]. Available at: <http://arxiv.org/abs/1906.03351> (Accessed: 13 November 2024).

**Best Practices for Addressing New Challenges in Testing and Evaluating Artificial Intelligence Enabled Systems (n.d.)** The Acquisition Innovation Research Center. Available at: <https://acqirc.org/publications/perspectives/best-practices-for-addressing-new-challenges-in-testing-and-evaluating-artificial-intelligence-enabled-systems/> (Accessed: 4 December 2024).

**Bird, S., Klein, E. and Loper, E. (2024)** *NLTK Book* [online]. Available at: <https://www.nltk.org/book/> (Accessed: 26 November 2024).

**Black Box Testing: An In-depth Tutorial with Examples and Techniques (2007)** Available at: <https://www.softwaretestinghelp.com/black-box-testing/> (Accessed: 4 December 2024).

**Braiek, H.B. and Khomh, F. (2018)** ‘On Testing Machine Learning Programs’, *arXiv* [Preprint]. Available at: <https://doi.org/10.48550/arXiv.1812.02257>.

**Brownlee, J. (2024a)** *A Gentle Introduction to Imbalanced Classification - MachineLearningMastery.Com* [online]. Available at: <https://machinelearningmastery.com/what-is-imbalanced-classification/?utm_source=chatgpt.com> (Accessed: 26 November 2024).

**Brownlee, J. (2024b)** *Imbalanced Classification with Python* [online]. Available at: <https://machinelearningmastery.com/imbalanced-classification-with-python/?utm_source=chatgpt.com> (Accessed: 26 November 2024).

**Brownlee, J. (2024c)** *SMOTE for Imbalanced Classification with Python - MachineLearningMastery.Com* [online]. Available at: <https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/?utm_source=chatgpt.com> (Accessed: 26 November 2024).

**Camacho-Collados, J. and Pilehvar, M.T. (2018)** ‘On the Role of Text Preprocessing in Neural Network Architectures: An Evaluation Study on Text Categorization and Sentiment Analysis’, *arXiv* [Preprint]. Available at: <https://doi.org/10.48550/arXiv.1707.01780>.

**Casper, S., Ezell, C., Siegmann, C., Kolt, N., Curtis, T.L., Bucknall, B., Haupt, A., Wei, K., Scheurer, J., Hobbhahn, M. et al. (2024)** ‘Black-Box Access is Insufficient for Rigorous AI Audits’, in *The 2024 ACM Conference on Fairness, Accountability, and Transparency*, pp. 2254–2272. Available at: <https://doi.org/10.1145/3630106.3659037>.

**Chakraborty, S., Bedi, A.S., Zhu, S., An, B., Manocha, D. and Huang, F. (2023)** ‘On the Possibilities of AI-Generated Text Detection’, *arXiv* [Preprint]. Available at: <https://doi.org/10.48550/arXiv.2304.04736>.

**Chandana, I., Reshma, O.M., Sree, N.G., Reddy, B.J. and Shareefunnisa, S. (2024)** ‘Detecting AI Generated Text’, in *2024 2nd World Conference on Communication & Computing (WCONF)*. IEEE, pp. 1–6. Available at: <https://doi.org/10.1109/WCONF61366.2024.10692028>.

**ChatGPT [online] (2024)** Available at: <https://chatgpt.com/?ref=dotcom> (Accessed: 26 November 2024).

**Chen, T., Hirota, Y., Otani, M., Garcia, N. and Nakashima, Y. (2024)** ‘Would Deep Generative Models Amplify Bias in Future Models?’, in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10833–10843. Available at: <https://doi.org/10.1109/CVPR52733.2024.01030>.

**Choi, J.D., Tetreault, J. and Stent, A. (2015)** ‘It Depends: Dependency Parser Comparison Using A Web-based Evaluation Tool’, in Zong, C. and Strube, M. (eds.) *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China: Association for Computational Linguistics, pp. 387–396. Available at: <https://doi.org/10.3115/v1/P15-1038>.

**Chopra, R. (2018)** ‘Chapter 4: White-Box (or Structural) Testing Techniques’, in *Software Testing: A Self-Teaching Introduction*. Mercury Learning and Information, pp. 133–206. Available at: <https://doi.org/10.1515/9781683923169-004>.

**Dathathri, S., See, A., Ghaisas, S., Huang, P.-S., McAdam, R., Welbl, J. et al. (2024)** ‘Scalable watermarking for identifying large language model outputs’, *Nature*, 634(8035), pp. 818–823. Available at: <https://doi.org/10.1038/s41586-024-08025-4>.

**Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K. (2019)** ‘BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding’, *arXiv* [Preprint]. Available at: <https://doi.org/10.48550/arXiv.1810.04805>.

**Dias, T., Batista, A., Maia, E. and Praça, I. (2023)** ‘TestLab: An Intelligent Automated Software Testing Framework’, *arXiv* [Preprint]. Available at: <https://doi.org/10.48550/arXiv.2306.03602>.

**Eke, D.O. (2023)** ‘ChatGPT and the rise of generative AI: Threat to academic integrity?’, *Journal of Responsible Technology*, 13, 100060. Available at: <https://doi.org/10.1016/j.jrt.2023.100060>.

**Elkhatat, A.M., Elsaid, K. and Almeer, S. (2023)** ‘Evaluating the efficacy of AI content detection tools in differentiating between human and AI-generated text’, *International Journal for Educational Integrity*, 19(1), pp. 1–16. Available at: <https://doi.org/10.1007/s40979-023-00140-5>.

**Gehrmann, S., Strobelt, H. and Rush, A.M. (2019)** ‘GLTR: Statistical Detection and Visualization of Generated Text’, *arXiv* [Preprint]. Available at: <https://doi.org/10.48550/arXiv.1906.04043>.

**Géron, A. (2022)** *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow*. 3rd edn. O’Reilly Media Inc.

**GitHub - Ertis-Research/Kafka-Ml: Kafka-ML: Connecting the Data Stream with ML/AI Frameworks (2024)** Available at: <https://github.com/ertis-research/kafka-ml> (Accessed: 26 November 2024).

**GPT-4 [online] (2024)** Available at: <https://openai.com/index/gpt-4-research/> (Accessed: 5 December 2024).

**Guillén-Yparrea, N. and Hernández-Rodríguez, F. (2024)** ‘Unveiling Generative AI in Higher Education: Insights from Engineering Students and Professors’, in *2024 IEEE Global Engineering Education Conference (EDUCON)*. IEEE, pp. 1–5. Available at: <https://doi.org/10.1109/EDUCON60312.2024.10578876>.

**Guo, X., Zhang, S., He, Y., Zhang, T., Feng, W., Huang, H. and Ma, C. (2024)** ‘DeTeCtive: Detecting AI-generated Text via Multi-Level Contrastive Learning’, *arXiv* [Preprint]. Available at: <http://arxiv.org/abs/2410.20964> (Accessed: 13 November 2024).

**Hamilton, T. (2024a)** *Boundary Value Analysis and Equivalence Partitioning* [online]. Available at: <https://www.guru99.com/equivalence-partitioning-boundary-value-analysis.html> (Accessed: 4 December 2024).

**Hamilton, T. (2024b)** *White Box Testing – What Is, Techniques, Example & Types* [online]. Available at: <https://www.guru99.com/white-box-testing.html> (Accessed: 5 December 2024).

**Hardeniya, N., Perkins, J., Chopra, D., Joshi, N. and Mathur, I. (2016)** *Natural Language Processing: Python and NLTK*. Packt Publishing Ltd.

**Hassanipour, S., Nayak, S., Bozorgi, A. et al. (2024)** ‘The Ability of ChatGPT in Paraphrasing Texts and Reducing Plagiarism: A Descriptive Analysis’, *JMIR Medical Education*, 10(1), e53308. Available at: <https://doi.org/10.2196/53308>.

**Hassija, V., Chamola, V., Mahapatra, A., Singal, A., Goel, D., Huang, K., Scardapane, S., Spinelli, I., Mahmud, M. and Hussain, A. (2024)** ‘Interpreting Black-Box Models: A Review on Explainable Artificial Intelligence’, *Cognitive Computation*, 16(1), pp. 45–74. Available at: <https://doi.org/10.1007/s12559-023-10179-8>.

**Hendrik Kirchner, J., Ahmad, L., Aaronson, S. and Leike, J. (2023)** ‘New AI Classifier for Indicating AI-Written Text’ [online]. OpenAI. Available at: <https://openai.com/index/new-ai-classifier-for-indicating-ai-written-text/> (Accessed: 14 November 2024).

**Holtzman, A., Buys, J., Du, L., Forbes, M. and Choi, Y. (2020)** ‘The Curious Case of Neural Text Degeneration’, *arXiv* [Preprint]. Available at: <https://doi.org/10.48550/arXiv.1904.09751>.

**Jawahar, G., Sagot, B. and Seddah, D. (2019)** ‘What Does BERT Learn about the Structure of Language?’, in Korhonen, A., Traum, D. and Màrquez, L. (eds.) *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, pp. 3651–3657. Available at: <https://doi.org/10.18653/v1/P19-1356>.

**Joshi, P.D., Pocker, S., Dandekar, R.A., Dandekar, R. and Panat, S. (2024)** ‘HULLMI: Human vs LLM identification with explainability’, *arXiv* [Preprint]. Available at: <http://arxiv.org/abs/2409.04808> (Accessed: 20 November 2024).

**Kalmykov, V.L. and Kalmykov, L.V. (2024)** ‘XXAI: Towards eXplicitly eXplainable Artificial Intelligence’, *arXiv* [Preprint]. Available at: <https://doi.org/10.48550/arXiv.2401.03093>.

**Kaushal, A. and Mahowald, K. (2022)** ‘What do tokens know about their characters and how do they know it?’, *arXiv* [Preprint]. Available at: <https://doi.org/10.48550/arXiv.2206.02608>.

**Knowledge extraction (2024)** *Wikipedia*. Available at: <https://en.wikipedia.org/w/index.php?title=Knowledge_extraction&oldid=1250059280> (Accessed: 26 November 2024).

**Kramp, S., Cassani, G. and Emmery, C. (2023)** ‘Native Language Identification with Big Bird Embeddings’, *arXiv* [Preprint]. Available at: <http://arxiv.org/abs/2309.06923> (Accessed: 14 November 2024).

**Kumar, S. (2021)** ‘Metrics to Evaluate your Classification Model to Take the Right Decisions’, *Analytics Vidhya*. Available at: <https://www.analyticsvidhya.com/blog/2021/07/metrics-to-evaluate-your-classification-model-to-take-the-right-decisions/> (Accessed: 26 November 2024).

**Kumarage, T., Bhattacharjee, A., Padejski, D., Roschke, K., Gillmor, D., Ruston, S. et al. (2023)** ‘J-Guard: Journalism Guided Adversarially Robust Detection of AI-generated News’, *arXiv* [Preprint]. Available at: <http://arxiv.org/abs/2309.03164> (Accessed: 13 November 2024).

**Kuo, C. (2023)** *The Handbook of NLP with Gensim: Leverage Topic Modeling to Uncover Hidden Patterns, Themes, and Valuable Insights within Textual Data*. Packt Publishing Ltd.

**Kurt Pehlivanoğlu, M., Gobosho, R.T., Syakura, M.A., Shanmuganathan, V. and de-la-Fuente-Valentín, L. (2024)** ‘Comparative analysis of paraphrasing performance of ChatGPT, GPT-3, and T5 language models using a new ChatGPT generated dataset: ParaGPT’, *Expert Systems*, 41(11), e13699. Available at: <https://doi.org/10.1111/exsy.13699>.

**Leskovec, J., Rajaraman, A. and Ullman, J.D. (2014)** *Mining of Massive Datasets*. Cambridge University Press.

**Li, Y., Li, Q., Cui, L., Bi, W., Wang, Z., Wang, L., Yang, L., Shi, S. and Zhang, Y. (2024)** ‘MAGE: Machine-generated Text Detection in the Wild’, in Ku, L.-W., Martins, A. and Srikumar, V. (eds.) *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Bangkok, Thailand: ACL, pp. 36–53. Available at: <https://doi.org/10.18653/v1/2024.acl-long.3>.

**Li, Z., Jiang, X., Shang, L. and Liu, Q. (2019)** ‘Decomposable Neural Paraphrase Generation’, *arXiv* [Preprint]. Available at: <https://doi.org/10.48550/arXiv.1906.09741>.

**Lin, Z., Gou, Z., Gong, Y., Liu, X., Shen, Y., Xu, R. et al. (2024)** ‘Rho-1: Not All Tokens Are What You Need’, *arXiv* [Preprint]. Available at: <https://doi.org/10.48550/arXiv.2404.07965>.

**Load Testing vs Stress Testing vs Performance Testing [online] (2024)** Available at: <https://www.guru99.com/performance-vs-load-vs-stress-testing.html> (Accessed: 5 December 2024).

**Loria, S. (n.d.)** ‘TextBlob Documentation’. Available at: <https://textblob.readthedocs.io/> (no explicit access date given).

**Manish (2020)** ‘Quick Guide to Evaluation Metrics for Supervised and Unsupervised Machine Learning’, *Analytics Vidhya*. Available at: <https://www.analyticsvidhya.com/blog/2020/10/quick-guide-to-evaluation-metrics-for-supervised-and-unsupervised-machine-learning/> (Accessed: 26 November 2024).

**McKinney, W. (2018)** *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython*. 2nd edn. O’Reilly Media.

**Mitchell, E., Lee, Y., Khazatsky, A., Manning, C.D. and Finn, C. (2023)** ‘DetectGPT: Zero-Shot Machine-Generated Text Detection Using Probability Curvature’, *arXiv* [Preprint]. Available at: <http://arxiv.org/abs/2301.11305> (Accessed: 13 November 2024).

**Montero, I., Pappas, N. and Smith, N.A. (2021)** ‘Sentence Bottleneck Autoencoders from Transformer Language Models’, *arXiv* [Preprint]. Available at: <http://arxiv.org/abs/2109.00055> (Accessed: 14 November 2024).

**Naidu, G., Zuva, T. and Sibanda, E.M. (2023)** ‘A Review of Evaluation Metrics in Machine Learning Algorithms’, in Silhavy, R. and Silhavy, P. (eds.) *Artificial Intelligence Application in Networks and Systems*. Cham: Springer International Publishing, pp. 15–25. Available at: <https://doi.org/10.1007/978-3-031-35314-7_2>.

**NLP Tutorial – Text Pre-Processing Techniques for Beginners [online] (2023)** freeCodeCamp.org. Available at: <https://www.freecodecamp.org/news/natural-language-processing-techniques-for-beginners/> (Accessed: 26 November 2024).

**OpenAI, Achiam, J., Adler, S. et al. (2024)** ‘GPT-4 Technical Report’, *arXiv* [Preprint]. Available at: <https://doi.org/10.48550/arXiv.2303.08774>.

**Owusu-Adjei, M., Hayfron-Acquah, J.B., Frimpong, T. and Abdul-Salaam, G. (2023)** ‘Imbalanced class distribution and performance evaluation metrics: A systematic review of prediction accuracy for determining model performance in healthcare systems’, *PLOS Digital Health*, 2(11), e0000290. Available at: <https://doi.org/10.1371/journal.pdig.0000290>.

**Paaß, G. and Giesselbach, S. (2023a)** ‘Improving Pre-trained Language Models’, in *Foundation Models for Natural Language Processing: Pre-Trained Language Models Integrating Media*. Cham: Springer International Publishing, pp. 79–159. Available at: <https://doi.org/10.1007/978-3-031-23190-2_3>.

**Paaß, G. and Giesselbach, S. (2023b)** ‘Pre-trained Language Models’, in *Foundation Models for Natural Language Processing: Pre-Trained Language Models Integrating Media*. Cham: Springer International Publishing, pp. 19–78. Available at: <https://doi.org/10.1007/978-3-031-23190-2_2>.

**Pandas Documentation — Pandas 2.2.3 Documentation [online] (2024)** Available at: <https://pandas.pydata.org/docs/> (Accessed: 26 November 2024).

**Parashchenko, R. (2024)** ‘Stress Testing, Its Goal and Benefits’, *StrongQA - QA Portal*. Available at: <https://strongqa.com/qa-portal/knowledge-base/testing-types/performance-testing/stress-testing> (Accessed: 5 December 2024).

**(PDF) Data Structures for Statistical Computing in Python [online] (2024)** Available at: <https://www.researchgate.net/publication/265001241_Data_Structures_for_Statistical_Computing_in_Python> (Accessed: 5 December 2024).

**Pei, K., Cao, Y., Yang, J. and Jana, S. (2017)** ‘DeepXplore: Automated Whitebox Testing of Deep Learning Systems’, in *Proceedings of the 26th Symposium on Operating Systems Principles*, pp. 1–18. Available at: <https://doi.org/10.1145/3132747.3132785>.

**Petrov, A., Malfa, E.L., Torr, P.H.S. and Bibi, A. (2023)** ‘Language Model Tokenizers Introduce Unfairness Between Languages’, *arXiv* [Preprint]. Available at: <https://doi.org/10.48550/arXiv.2305.15425>.

**Proposal for a REGULATION OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL … (2024)** Laying down harmonised rules on Artificial Intelligence (AI Act). Available at: <https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:52021PC0206> (Accessed: 5 December 2024).

**Prova, N. (2024)** ‘Detecting AI Generated Text Based on NLP and Machine Learning Approaches’, *arXiv* [Preprint]. Available at: <http://arxiv.org/abs/2404.10032> (Accessed: 13 November 2024).

**Raptis, T.P. and Passarella, A. (2022)** ‘On Efficiently Partitioning a Topic in Apache Kafka’, in *2022 International Conference on Computer, Information and Telecommunication Systems (CITS)*. IEEE, pp. 1–8. Available at: <https://doi.org/10.1109/CITS55221.2022.9832981>.

**Ribeiro, M.T., Singh, S. and Guestrin, C. (2016)** ‘“Why Should I Trust You?”: Explaining the Predictions of Any Classifier’, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1135–1144. Available at: <https://doi.org/10.1145/2939672.2939778>.

**Robust Machine Learning on Streaming Data Using Kafka and Tensorflow-IO | TensorFlow I/O [online] (2024)** TensorFlow. Available at: <https://www.tensorflow.org/io/tutorials/kafka> (Accessed: 26 November 2024).

**Roman, A. (2018)** ‘Black-Box Testing Techniques’, in *A Study Guide to the ISTQB® Foundation Level 2018 Syllabus: Test Techniques and Sample Mock Exams*. Cham: Springer International Publishing, pp. 25–60. Available at: <https://doi.org/10.1007/978-3-319-98740-8_4>.

**Sadasivan, V.S., Kumar, A., Balasubramanian, S., Wang, W. and Feizi, S. (2023)** ‘Can AI-Generated Text be Reliably Detected?’, *arXiv* [Preprint]. Available at: <http://arxiv.org/abs/2303.11156> (Accessed: 13 November 2024).

**Sahil (2020)** ‘A Tour of Evaluation Metrics for Machine Learning’, *Analytics Vidhya*. Available at: <https://www.analyticsvidhya.com/blog/2020/11/a-tour-of-evaluation-metrics-for-machine-learning/> (Accessed: 26 November 2024).

**Saito, T. and Rehmsmeier, M. (2015)** ‘The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets’, *PLOS ONE*, 10(3), e0118432. Available at: <https://doi.org/10.1371/journal.pone.0118432>.

**Salloum, S., Gaber, T., Vadera, S. and Shaalan, K. (2022)** ‘A Systematic Literature Review on Phishing Email Detection Using Natural Language Processing Techniques’, *IEEE Access*, 10, pp. 65703–65727. Available at: <https://doi.org/10.1109/ACCESS.2022.3183083>.

**Scikit-Learn: Machine Learning in Python — Scikit-Learn 1.5.2 Documentation [online] (2024)** Available at: <https://scikit-learn.org/stable/> (Accessed: 5 December 2024).

**Sevetlidis, V., Pavlidis, G., Mouroutsos, S.G. and Gasteratos, A. (2024)** ‘Dense-PU: Learning a Density-Based Boundary for Positive and Unlabeled Learning’, *IEEE Access*, 12, pp. 90287–90298. Available at: <https://doi.org/10.1109/ACCESS.2024.3420453>.

**Şevgi̇n, H. (2023)** ‘A comparative study of ensemble methods in the field of education: Bagging and Boosting algorithms’, *International Journal of Assessment Tools in Education*, 10(3), pp. 544–562. Available at: <https://doi.org/10.21449/ijate.1167705>.

**Shao, R., Schwarz, R., Clifton, C. and Delp, E. (2024)** ‘A Natural Approach for Synthetic Short-Form Text Analysis’, in Calzolari, N. et al. (eds.) *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*. Torino, Italy: ELRA and ICCL, pp. 1034–1042. Available at: <https://aclanthology.org/2024.lrec-main.92> (Accessed: 13 November 2024).

**Shi, Y., Sheng, Q., Cao, J., Mi, H., Hu, B. and Wang, D. (2024)** ‘Ten Words Only Still Help: Improving Black-Box AI-Generated Text Detection via Proxy-Guided Efficient Re-Sampling’, in *Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 494–502. Available at: <https://doi.org/10.24963/ijcai.2024/55>.

**Shu, K., Wang, S. and Liu, H. (2019)** ‘Beyond News Contents: The Role of Social Context for Fake News Detection’, in *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining (WSDM ’19)*. Melbourne, VIC: ACM, pp. 312–320. Available at: <https://doi.org/10.1145/3289600.3290994>.

**Software Testing - Boundary Value Analysis vs Equivalence Partitioning - GeeksforGeeks [online] (2024)** Available at: <https://www.geeksforgeeks.org/software-testing-boundary-value-analysis-vs-equivalence-partitioning/> (Accessed: 4 December 2024).

**Software Testing - Boundary Value Analysis vs Equivalence Partitioning [online] (2022)** *GeeksforGeeks*. Available at: <https://www.geeksforgeeks.org/software-testing-boundary-value-analysis-vs-equivalence-partitioning/> (Accessed: 5 December 2024).

**spaCy · Industrial-Strength Natural Language Processing in Python [online] (2024a)** Available at: <https://spacy.io/> (Accessed: 26 November 2024).

**spaCy · Industrial-Strength Natural Language Processing in Python [online] (2024b)** Available at: <https://spacy.io/> (Accessed: 5 December 2024).

**Srivastava, T. (2024)** ‘12 Important Model Evaluation Metrics for Machine Learning Everyone Should Know (Updated 2024)’, *Analytics Vidhya*. Available at: <https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/> (Accessed: 26 November 2024).

**stephen-sumner (2023)** ‘Recommendations for Performance Testing - Microsoft Azure Well-Architected Framework’ [online]. Microsoft. Available at: <https://learn.microsoft.com/en-us/azure/well-architected/performance-efficiency/performance-test> (Accessed: 5 December 2024).

**Stray, J. (2023)** ‘The AI Learns to Lie to Please You: Preventing Biased Feedback Loops in Machine-Assisted Intelligence Analysis’, *Analytics*, 2(2), pp. 350–358. Available at: <https://doi.org/10.3390/analytics2020020>.

**Tarek Amr (2020)** *Hands-On Machine Learning with Scikit-Learn and Scientific Python Toolkits: A Practical Guide to Implementing Supervised and Unsupervised Machine Learning Algorithms in Python*. Packt Publishing. Available at: <https://search.ebscohost.com/login.aspx?direct=true&AuthType=sso&db=nlebk&AN=2562942&site=eds-live&scope=site> (Accessed: 26 November 2024).

**Text Preprocessing in NLP [online] (2024)** *GeeksforGeeks*. Available at: <https://www.geeksforgeeks.org/text-preprocessing-for-nlp-tasks/> (Accessed: 26 November 2024).

**Tian, Y., Chen, H., Wang, X., Bai, Z., Zhang, Q., Li, R. et al. (2023)** ‘Multiscale Positive-Unlabeled Detection of AI-Generated Texts’, *arXiv* [Preprint]. Available at: <http://arxiv.org/abs/2305.18149> (Accessed: 13 November 2024).

**Tian, Y., Pan, Z. and Peng, N. (2024)** ‘Detecting Machine-Generated Long-Form Content with Latent-Space Variables’, *arXiv* [Preprint]. Available at: <http://arxiv.org/abs/2410.03856> (Accessed: 13 November 2024).

**Tithi (2021)** ‘Essential Text Pre-processing Techniques for NLP!’, *Analytics Vidhya*. Available at: <https://www.analyticsvidhya.com/blog/2021/09/essential-text-pre-processing-techniques-for-nlp/> (Accessed: 26 November 2024).

**Uzun, L. (2023)** ‘ChatGPT and Academic Integrity Concerns: Detecting Artificial Intelligence Generated Content’, *Language Education and Technology*, 3(1). Available at: <https://www.langedutech.com/letjournal/index.php/let/article/view/49> (Accessed: 13 November 2024).

**Vasiliev, Y. (2020)** *Natural Language Processing with Python and spaCy: A Practical Introduction*. No Starch Press.

**Waehner, K. (2020)** ‘Streaming Machine Learning with Kafka-native Model Deployment’, *Kai Waehner Blog*. Available at: <https://www.kai-waehner.de/blog/2020/10/27/streaming-machine-learning-kafka-native-model-server-deployment-rpc-embedded-streams/> (Accessed: 26 November 2024).

**Wahle, J.P., Gipp, B. and Ruas, T. (2024)** ‘Paraphrase Types for Generation and Detection’, *arXiv* [Preprint]. Available at: <https://doi.org/10.48550/arXiv.2310.14863>.

**Wani, M.A., ElAffendi, M. and Shakil, K.A. (2024)** ‘AI-Generated Spam Review Detection Framework with Deep Learning Algorithms and Natural Language Processing’, *Computers*, 13(10), 264. Available at: <https://doi.org/10.3390/computers13100264>.

**Weber-Wulff, D., Anohina-Naumeca, A., Bjelobaba, S. et al. (2023)** ‘Testing of detection tools for AI-generated text’, *International Journal for Educational Integrity*, 19(1), pp. 1–39. Available at: <https://doi.org/10.1007/s40979-023-00146-z>.

**White Box Testing - Software Engineering [online] (2018)** *GeeksforGeeks*. Available at: <https://www.geeksforgeeks.org/software-engineering-white-box-testing/> (Accessed: 5 December 2024).

**Wiesbrock, H.-W. and Großmann, J. (2024)** ‘Outline of an Independent Systematic Blackbox Test for ML-based Systems’, *arXiv* [Preprint]. Available at: <https://doi.org/10.48550/arXiv.2401.17062>.

**Yeturu, J., Elango, P., Raja, S. and Kumar, P. (2023)** ‘A Novel Ensemble Stacking Classification of Genetic Variations Using Machine Learning Algorithms’.

**Zagorulko, D.I. (2023)** ‘CHATGPT IN NEWSROOMS: Adherence of AI-generated content to journalism standards and prospects for its implementation in digital media’, *Scientific notes of V. I. Vernadsky Taurida National University*, Series: Philology. Journalism, 2(1), pp. 319–325. Available at: <https://doi.org/10.32782/2710-4656/2023.1.2/50>.

**Zellers, R., Holtzman, A., Rashkin, H., Bisk, Y., Farhadi, A., Roesner, F. and Choi, Y. (2020)** ‘Defending Against Neural Fake News’, *arXiv* [Preprint]. Available at: <http://arxiv.org/abs/1905.12616> (Accessed: 13 November 2024).

**Zhang, Y., Leng, Q., Zhu, M., Ding, R., Wu, Y., Song, J. and Gong, Y. (2024)** ‘Enhancing Text Authenticity: A Novel Hybrid Approach for AI-Generated Text Detection’, in *2024 IEEE 4th International Conference on Electronic Technology, Communication and Information (ICETCI)*. IEEE, pp. 433–438. Available at: <https://doi.org/10.1109/ICETCI61221.2024.10594194>.

**Zhou, J. and Bhat, S. (2021)** ‘Paraphrase Generation: A Survey of the State of the Art’, in Moens, M.-F. et al. (eds.) *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pp. 5075–5086. Available at: <https://doi.org/10.18653/v1/2021.emnlp-main.414>.

**Zhou, Y. and Wang, J. (2024)** ‘Detecting AI-Generated Texts in Cross-Domains’, in *Proceedings of the ACM Symposium on Document Engineering 2024*, pp. 1–4. Available at: <https://doi.org/10.1145/3685650.3685673>.

**Zimmermann, J., Champagne, L.E., Dickens, J.M. and Hazen, B.T. (2024)** ‘Approaches to improve preprocessing for Latent Dirichlet Allocation topic modeling’, *Decision Support Systems*, 185, 114310. Available at: <https://doi.org/10.1016/j.dss.2024.114310>.