universität wien

MASTERARBEIT | MASTER'S THESIS

Titel | Title

Hybrid Machine Learning Approaches for Detection of LLM-Generated English Texts

verfasst von | submitted by

Bohdan Zhvalevskyi BA

angestrebter akademischer Grad | in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien | Vienna, 2026

Studienkennzahl lt. Studienblatt | Degree programme code as it appears on the student record sheet:	UA 066 587
Studienrichtung lt. Studienblatt | Degree programme as it appears on the student record sheet:	Joint-Masterstudium Multilingual Technologies
Betreut von | Supervisor:	Assoz. Prof. Mag. Dr. Dagmar Gromann BSc

Acknowledgements

I would like to start this master's thesis by thanking my supervisor and mentor, Prof. Dagmar Gromann, who has been my constant source of guidance, support, and academic inspiration throughout my studies. Her expertise and feedback have been so very helpful to me, both during our classes and in the writing of this work. Thank you for pushing me to excel.

I am also thanking Liad Magen, who has shown me how interactive and engaging an academic learning experience can be, as well as for helping me awaken my interest for Machine Learning. Thank you for your support and encouragement, and for always having our cohort's back.

I am very grateful to my colleagues, my cohort, for their support and collaboration during our studies. I think that we made a great team and grew to be a close university family. I wish everyone of you the best of luck in your future endeavours, and I hope that we will stay in touch.

Perhaps the biggest thanks go to my family, to my mom and dad, my cousin. Everything I have achieved in this academic path would not have happened if it were not for their constant support and help. Thank you raising me to be who I am today, and for always being there for me. I am also thankful to my friends in Ukraine, especially to Olia, who have been a source of joy and support throughout my studies.

Abstract

Large Language Models (LLMs) have made machine-generated text common in education, journalism, and online communication, where its undisclosed use raises concerns about authenticity, authorship attribution, and academic integrity. Detectors that perform well on clean text often become unreliable when the text is adversarially manipulated through paraphrasing, synonym substitution, or style polishing. This master's thesis investigates whether adding extracted linguistic features to a neural classifier can improve the robustness of LLM-generated English text detection, and which of these features contribute most. To this end, it develops a hybrid architecture that fuses contextual representations from a pre-trained RoBERTa encoder with extracted linguistic features through a gated fusion mechanism, which weighs the two information sources differently for each individual input text. The model is trained and evaluated on the RAID subset of the RealBench benchmark, which spans multiple model families, domains, and adversarial attack types. The hybrid model reaches about 95% document-level F1, improving over a strong RoBERTa baseline by roughly 3.5 points, with the largest gains on the hardest attacks, namely style polishing and sentence-level paraphrasing. A permutation-based analysis identifies perplexity, lexical diversity, and syntactic complexity as the most informative linguistic features. The learned gate shows an interpretable, class-conditional pattern, relying more on linguistic features for human-written text and balancing both sources for LLM-generated text. Overall, the results suggest that linguistic features provide a complementary signal that partly offsets the fragility of neural detectors under adversarial pressure, though this robustness still depends on how well the training data covers the range of generators and attacks encountered in practice.

Zusammenfassung

Große Sprachmodelle (LLMs) haben maschinell generierte Texte in Bereichen wie Bildung, Journalismus und Online-Kommunikation alltäglich gemacht, wobei ihre nicht offengelegte Verwendung Bedenken bezüglich Authentizität, Urheberschaft und wissenschaftlicher Integrität aufwirft. Erkennungssysteme, die bei „sauberen“ Texten gute Ergebnisse liefern, werden oft unzuverlässig, wenn der Text durch Paraphrasierung, Synonymersetzung oder stilistische Überarbeitung gezielt manipuliert wird. Diese Masterarbeit untersucht, ob das Hinzufügen extrahierter linguistischer Merkmale zu einem neuronalen Klassifikator die Robustheit der Erkennung von durch LLMs generierten englischen Texten verbessern kann, und welche dieser Merkmale den größten Beitrag leisten. Zu diesem Zweck wird eine hybride Architektur entwickelt, die kontextuelle Repräsentationen aus einem vortrainierten RoBERTa-Encoder mit extrahierten linguistischen Merkmalen über einen „Gated Fusion“-Mechanismus kombiniert, der die beiden Informationsquellen für jeden einzelnen Eingabetext unterschiedlich gewichtet. Das Modell wird auf der RAID-Teilmenge des RealBench-Benchmarks trainiert und evaluiert, die mehrere Modellfamilien, Domänen und Arten von adversariellen Angriffen umfasst. Das Hybridmodell erreicht einen F1-Wert auf Dokumentebene von etwa 95%, was eine Verbesserung um rund 3,5 Punkte gegenüber einer starken RoBERTa-Baseline darstellt, wobei die größten Verbesserungen bei den schwierigsten Angriffen erzielt werden, nämlich bei der stilistischen Überarbeitung und der Paraphrasierung auf Satzebene. Eine permutationsbasierte Analyse identifiziert Perplexität, lexikalische Vielfalt und syntaktische Komplexität als die aussagekräftigsten linguistischen Merkmale. Das gelernte Gate zeigt ein interpretierbares, klassenabhängiges Verhalten, das sich bei von Menschen verfassten Texten stärker auf linguistische Merkmale stützt und bei von LLMs generierten Texten beide Quellen ausgewogen berücksichtigt. Insgesamt deuten die Ergebnisse darauf hin, dass linguistische Merkmale ein ergänzendes Signal liefern, das die Anfälligkeit neuronaler Detektoren unter adversariellem Druck teilweise ausgleicht, auch wenn diese Robustheit davon abhängt, wie gut die Trainingsdaten die in der Praxis auftretenden Generatoren und Angriffe abdecken.

Contents

List of Tables **11**

List of Figures **13**

1. Introduction **1**
1.1. Research Questions and Assumptions 2
1.2. Motivation and Objective 2

2. Machine Learning and Deep Learning **3**
2.1. Machine Learning Fundamentals 3
2.1.1. Supervised Learning and Classification 4
2.1.2. Unsupervised Learning 6
2.1.3. Self-Supervised Learning 8
2.1.4. Loss Functions and Optimization 9
2.1.5. Evaluation Metrics for Classification 12
2.1.6. Overfitting, Regularization, and Generalization 14
2.2. Deep Learning and Neural Networks 16
2.2.1. Neural Network Architectures 16

3. Transformer Architecture and Attention Mechanisms **21**
3.1. Attention Mechanisms 21
3.2. Self-Attention and Multi-Head Attention 23
3.3. Transformer Architecture Components 25
3.4. Positional Encodings and Input Representations 28
3.5. Encoder-Only, Decoder-Only, and Encoder-Decoder Variants 30
3.6. Gating Mechanisms and Dynamic Feature Selection 31

4. Natural Language Processing and Linguistic Features **35**
4.1. Text Representation and Tokenization 35
4.2. Word Embeddings and Distributional Semantics 36
4.3. Lexical Features and Vocabulary Statistics 37
4.4. Syntactic Features and Dependency Parsing 38
4.5. Discourse and Coherence Features 38
4.6. Stylometric Features 39

4.7. Language Model-Based Features 40
4.8. Perplexity and Probability-Based Metrics 40

5. Large Language Models **43**
5.1. Pre-Training Paradigms and Objectives 43
5.2. Transfer Learning and Fine-Tuning 44
5.3. Representative LLM Architectures and Models 45

6. Detection of LLM-Generated Text **49**
6.1. Problem Formulation and Detection Paradigms 49
6.2. Zero-Shot and Supervised Detection Methods 50
6.3. Feature-Based Detection Approaches 53
6.4. Watermarking and Provenance 54
6.5. Evaluation Metrics and Datasets 56
6.6. Adversarial Pressure on Detection 57
6.6.1. Taxonomy of Adversarial Attacks on Text 57
6.6.2. Character-Level and Word-Level Perturbations 58
6.6.3. Paraphrasing and Semantic Attacks 60
6.6.4. Perplexity Manipulation and Style Transfer 61
6.7. Adversarial Training and Defence Mechanisms 62
6.8. Robustness Evaluation Methodologies 63
6.9. Combining Neural and Feature-Based Approaches 64
6.9.1. Feature Fusion Strategies 65
6.9.2. Ensemble Methods and Model Combination 67
6.9.3. Pooling Strategies for Sequence Representations 68

7. Related Work **71**

8. Method **73**
8.1. Dataset 73
8.1.1. Data Pre-processing 75
8.1.2. Linguistic Feature Extraction 76
8.2. Model Architecture 78
8.3. Evaluation 80

9. Experimental Setting **83**
9.1. Training Configuration 83
9.2. Baseline and Ablation Study 84

10. Results **87**
10.1. Overall Model Performance 87
10.2. Robustness Across Attack Types 88

10.3. Performance Across LLMs 89
10.4. Error Analysis 90
10.5. Gating Mechanism Analysis 91
10.6. Feature Importance and Attention 91
10.7. Ablation Study 93

11. Discussion **95**

12. Conclusion **99**

Bibliography **101**

List of Tables

1. Attack types used, drawn from the RAID subset of RealBench . . . 74
2. Top 10 LLM models by sample amount in final selection 75
3. Final set of 25 linguistic features organized by category 77
4. Complete hyperparameter configuration for the hybrid model 84
5. Performance comparison on RAID test set 88
6. Document-level robustness by RAID attack type 88
7. Document-level detection rate by LLM, reported as a percentage . . 90
8. Learned gate values for test set samples 91
9. Top ten features by permutation importance, with each feature's attention rank 92
10. Fusion ablation study at the document level 93

List of Figures

1. Architecture of the proposed hybrid RoBERTa model 79

1. Introduction

Large Language Models (LLMs), such as GPT-3 (Brown et al., 2020) and GPT-4, have become widely accessible to users since late 2022. As a result, digital platforms, academic papers, and online communication increasingly contain text that may have been generated or modified by LLMs, often without disclosure (Wu et al., 2025). This development raises concerns in domains where authenticity and source attribution are essential, particularly in education, journalism, and academic writing.

While early detection methods achieved high accuracy on clean datasets, they often fail when users apply adversarial techniques to disguise LLM-generated content. Attacks such as paraphrasing, synonym substitution, and style polishing can make detection substantially more difficult (Dugan et al., 2024; He et al., 2025). This problem becomes even more challenging as modern LLMs produce text that closely resembles natural human writing. LLM-generated text is vulnerable to adversarial manipulation that targets both statistical and learned detection signatures (He et al., 2024b).

Recent studies (Knudsen and Hardmeier, 2025; Yadagiri et al., 2024) suggest that combining Transformer-based representations with explicit linguistic features may improve detection robustness. However, most existing hybrid approaches have been evaluated primarily on non-adversarial datasets that may contain trivial stylistic artefacts, leaving their performance under realistic attack conditions unclear. This master's thesis investigates the effectiveness of including linguistic features in the detection process and identifies which features could potentially contribute most to detecting LLM-generated content. To this end, the work develops and evaluates a hybrid detection architecture that combines the representational power of pre-trained Transformer encoders with computational stylometric features through a gated fusion mechanism that adaptively weighs the contribution of each information source based on sample characteristics. The code is available on GitHub[1].

[1] `https://github.com/foggyghost0/Hybrid-Machine-Learning-Approaches-for-Detection-of-LLM-Generated-Texts`

1.1. Research Questions and Assumptions

This master's thesis investigates the following central research questions: To what extent can a hybrid architecture that fuses Transformer-based embeddings with computational stylometric features improve robust detection of LLM-generated English text under adversarial conditions? Which specific linguistic features contribute most to detection performance and robustness in this context?

The research is based on several key assumptions. First, LLM-generated text exhibits measurable statistical and stylistic differences from human writing, even after adversarial manipulation. Second, Transformer encoders capture global contextual patterns and semantic coherence, while explicit linguistic features encode properties such as syntactic complexity, lexical diversity, and discourse structure that Transformers may only partially learn. Third, a gated fusion mechanism can dynamically balance these two information sources based on input characteristics, providing better robustness than fixed combination strategies. Finally, adversarial transformations affect Transformer representations and linguistic features differently, making their combination more resilient than either approach alone.

1.2. Motivation and Objective

The motivation for this master's thesis stems from the practical need for reliable detection systems that remain effective under realistic adversarial conditions. Current Transformer-based detectors achieve high accuracy on clean data but often show substantial performance degradation when tested on adversarially modified text. Similarly, zero-shot methods that rely on statistical signatures are vulnerable to attacks that specifically target these signatures.

The research should provide interpretable insights into which linguistic features contribute most to detection under adversarial conditions. This is relevant for researchers developing robust detection systems, educators seeking tools to identify LLM-generated submissions, and platform operators working to maintain content authenticity.

2. Machine Learning and Deep Learning

This chapter establishes the theoretical foundations of machine learning and deep learning that underpin the modelling choices made throughout this master's thesis. It begins with the core principles of machine learning, covering supervised learning and classification, unsupervised learning, self-supervised learning and pre-training, loss functions and optimization, evaluation metrics, and generalization. It then proceeds to deep learning, examining neural network architectures, and training algorithms. Together, these topics provide the basis and conceptual framework needed to understand the proposed detection architecture in subsequent chapters.

2.1. Machine Learning Fundamentals

Machine learning is the study of computer algorithms that improve their performance on a task through experience, rather than through explicit programming (Mitchell, 1997). At its core, a machine learning system learns a mapping from inputs to outputs by seeing examples, adjusting its internal parameters until its predictions align closely with the observed data. This stands in contrast to classical rule-based systems, where a human expert manually encodes the decision logic. The shift toward learned representations has proven particularly powerful in domains where the underlying rules are too complex, too numerous, too high-level or simply too poorly understood by humans to write down by hand (Goodfellow et al., 2016).

Within the broader field, the problems solved by machine learning algorithms are typically grouped into several paradigms, as seen by Bishop and Nasrabadi (2006): supervised learning, unsupervised learning, self-supervised learning, and reinforcement learning. Supervised learning, which is the paradigm most relevant to this master's thesis, involves training a model on a labelled dataset, where each input example is paired with a known output. Unsupervised learning, by contrast, seeks to uncover structure in data without any labels, while self-supervised learning derives supervisory signals from the data itself and is discussed in more detail later in this chapter. Reinforcement learning trains an agent to maximize a reward signal through interaction with an environment, but it is not considered further here because it does not form part of the modelling framework used in this master's

thesis.

2.1.1. Supervised Learning and Classification

Supervised learning is the machine learning paradigm in which a model is trained on a labelled dataset consisting of input-output data pairs $\{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$, where each input $\mathbf{x}_i \in \mathcal{X}$ is associated with a known output $y_i \in \mathcal{Y}$. The goal is to learn a mapping $f : \mathcal{X} \to \mathcal{Y}$ that generalizes well to unseen inputs drawn from the same underlying distribution (Goodfellow et al., 2016; Bishop and Nasrabadi, 2006). Depending on the structure of the output space $\mathcal{Y}$, supervised problems are conventionally divided into two tasks. In regression, $\mathcal{Y}$ is continuous, and the model predicts a real-valued quantity such as a price or a temperature. In classification, $\mathcal{Y}$ is a discrete set of class labels and the model assigns each input to one of a finite number of predefined categories (Bishop and Nasrabadi, 2006; Hastie et al., 2009). The work in this master's thesis is framed as a binary classification problem, in which $\mathcal{Y} = \{0, 1\}$, although the broader discussion below applies to multi-class settings $\mathcal{Y} = \{1, 2, \ldots, k\}$ as well.

A variety of algorithm families have been developed within this paradigm, each making different assumptions about how the mapping f should be parameterized (Bishop and Nasrabadi, 2006; Hastie et al., 2009). Logistic regression models the conditional class probability directly as a sigmoid of a linear combination of features. Naive Bayes classifiers apply Bayes' rule under a conditional independence assumption over the input features. Support vector machines look for the separating boundary that maximizes the margin to the nearest training examples, with kernels providing access to non-linear decision boundaries. Decision trees recursively partition the input space into axis-aligned regions, and ensemble methods such as random forests and gradient-boosted trees combine many such trees to produce stronger predictors. k-nearest-neighbour classification predicts a label by majority vote over the closest training examples in feature space. Finally, neural networks, which underpin the architectures developed later in this master's thesis, parameterize f as the composition of many learned non-linear transformations.

The structure of the hypothesis class $\mathcal{H}$ that each of these families spans shapes both what the model can express and how it generalizes. In practice, learning is constrained to such a hypothesis class, defined as the family of mappings the model is capable of expressing given its architecture and parameterization. The choice of $\mathcal{H}$ encodes what are called inductive biases, which are assumptions about the structure of the problem that guide the model toward certain solutions over others (Shalev-Shwartz and Ben-David, 2014). A linear classifier, for example, restricts $\mathcal{H}$ to mappings that produce a linear decision surface in input space. This does not mean that the data must be linearly separable for the model to be trainable; given input that is not linearly separable, a linear classifier will still converge on

the best-fitting linear boundary under its objective, but its representational ceiling means that no setting of its parameters can recover a non-linear class structure (Bishop and Nasrabadi, 2006). Deep neural networks, by contrast, define a much richer hypothesis class through the composition of many non-linear transformations, allowing them to learn complex, hierarchical representations of the input. This expressive capacity is precisely what motivates their use in detection tasks, where the relationship between raw input features and the target label is rarely linear.

Classification can also be given a probabilistic interpretation that complements the geometric view above and clarifies what such a model is actually trying to estimate. Rather than directly predicting a hard label, a probabilistic classifier models the posterior distribution $P(y \mid \mathbf{x})$, that is, the probability of each class given the input, and then applies a decision rule to produce a final prediction. In the binary case, a common rule is to predict the positive class whenever $P(y = 1 \mid \mathbf{x}) > 0.5$, while in the multi-class case the predicted label is $\hat{y} = \arg\max_k P(y = k \mid \mathbf{x})$ (Bishop and Nasrabadi, 2006). This view applies to a wide range of classifiers, including logistic regression, Naive Bayes, and the softmax output of a neural network, all of which can be read as estimators of the same posterior under different parametric assumptions. The theoretical upper bound on classification performance is given by the Bayes-optimal classifier, which assigns each input to the class with the highest posterior probability under the true data-generating distribution (Murphy, 2012).

A fundamental principle in machine learning is that the performance of a model must be measured on data it has never seen during training. To correctly achieve this, the available labelled data is separated into three subsets, namely a training set, a validation set, and a test set (Goodfellow et al., 2016; Hastie et al., 2009). The training set is used to train the model, i.e., to adjust the weights of the model by minimizing the chosen loss function over the labelled examples. The validation set serves the main purpose of estimating the model's generalization behaviour on examples held out from training, and this estimate is used in turn to make modelling decisions that cannot be made on the training set alone, including the selection of hyperparameters, the choice between competing architectures, and the criterion for early stopping. Crucially, the test set is held back entirely until the model training has been conducted, at which point it is used to produce an unbiased estimate of the model's performance on genuinely unseen data (Goodfellow et al., 2016). Violating this protocol, for instance by using test set performance to guide any modelling choice, invalidates the final evaluation, because the reported performance will no longer reflect how the model would behave in deployment by generalizing on unseen data.

In practice, the proportion of data allocated to each split depends on the size of the dataset. Common ratios include 70/15/15 or 80/10/10 for training, validation,

and test sets respectively, though these are guidelines rather than fixed rules (Hastie et al., 2009). For large datasets, even a relatively small fraction reserved for validation and testing could be thousands of examples, which may be sufficient for reliable performance estimation. For smaller datasets, these fractions must be chosen carefully to ensure that the validation and test sets are large enough to give statistically meaningful results. A particularly important concern in this context is data leakage, which occurs when information from the validation or test set influences the training process (Hastie et al., 2009). Common causes include preprocessing steps, such as feature normalization, that are computed over the full dataset before splitting, or the use of test set results to select among models. Data leakage leads to over-optimistic reported performance and is one of the most common sources of irreproducible results in applied machine learning according to Goodfellow et al. (2016).

An additional consideration when constructing data splits is the distribution of class labels within each subset. In many detection tasks, the classes are not equally represented, since the event of interest may be rare relative to the background class, resulting in a skewed or imbalanced dataset. If examples are assigned to splits by simple random sampling, there is a risk that some subsets will contain very few instances of the minority class, which distorts both training and evaluation (Bishop and Nasrabadi, 2006). Stratified splitting addresses this by ensuring that each subset retains approximately the same class proportions as the full dataset. In the context of K-fold cross-validation, this becomes stratified K-fold cross-validation, where each fold is constructed to mirror the overall class distribution. Stratification is especially important for evaluation, because a test set that is not representative of the true class balance will produce metric estimates that do not reflect the conditions under which the model will be deployed in real life (Kohavi, 1995; Sechidis et al., 2011).

2.1.2. Unsupervised Learning

The unsupervised learning paradigm describes a broad family of methods that aim to discover structure in data without relying on manually created labels (Bishop and Nasrabadi, 2006; Hastie et al., 2009). In contrast to the supervised setting, unsupervised learning operates on a collection of input examples $\{(\mathbf{x}_i)\}_{i=1}^{N}$ alone, with no corresponding target variables. The central objective is to learn meaningful regularities in the underlying data distribution $p(\mathbf{x})$, and the specific form that this objective takes varies depending on the task at hand (Goodfellow et al., 2016). Unsupervised methods are not merely auxiliaries to supervised pipelines. They constitute a distinct family of techniques with their own purpose, applied whenever the goal is to characterize the data itself rather than to predict a known target (Hastie et al., 2009).

The principal categories of unsupervised methods correspond to the kinds of structure they aim to uncover. Clustering algorithms partition the data into groups of similar examples without reference to any label. k-means assigns each point to the nearest of k learned centroids and is the canonical example of this family. Hierarchical clustering builds a nested tree of partitions by repeatedly merging or splitting groups according to a chosen distance criterion. Density-based methods such as DBSCAN identify clusters as connected regions of high data density and treat low-density points as noise, which makes them appropriate when the number of groups is not known in advance and clusters may take non-convex shapes (Ester et al., 1996; Bishop and Nasrabadi, 2006). Dimensionality reduction methods aim to represent high-dimensional inputs in a lower-dimensional space that preserves the geometric or statistical properties most relevant to subsequent analysis. Principal Component Analysis (PCA) computes a linear projection that maximizes retained variance (Jolliffe, 2002), while non-linear techniques such as t-SNE and UMAP preserve local neighbourhood structure and are widely used for visualization. Autoencoders learn non-linear projections by training a neural network to reconstruct its input through a lower-dimensional bottleneck (Goodfellow et al., 2016). Density estimation methods, including Gaussian mixture models, kernel density estimators, and modern deep generative models, directly approximate the data distribution $p(\mathbf{x})$ and support tasks such as anomaly detection and synthetic data generation. Each of these categories provides a different lens on the same dataset, and which one is appropriate depends on the substantive question being asked of the data.

Unsupervised methods also play important supporting roles within supervised pipelines, providing tools that can improve the quality or efficiency of downstream learning (Goodfellow et al., 2016; LeCun et al., 2015). Dimensionality reduction through PCA or an autoencoder is routinely used as a preprocessing step to remove noise and reduce computational cost before a classifier is applied. Clustering can reveal structure in the unlabelled portion of a dataset, guiding the design of labelling strategies or providing features that complement the raw input representation. Historically, unsupervised pre-training, in which a network was first trained on a reconstruction or density-estimation objective and then fine-tuned on a supervised task, was one of the techniques that made the optimization of deep networks tractable before later advances in supervised training made it less necessary in many settings (Goodfellow et al., 2016).

The increasing use of objectives derived from the data itself, rather than from manual annotation, has gradually blurred the practical boundary between unsupervised and supervised learning. One outcome of this trend is the family of methods known as self-supervised learning, in which prediction targets are constructed automatically from the input data and a model is trained on the resulting objective

using the standard machinery of supervised optimization (LeCun et al., 2015; Goodfellow et al., 2016). This paradigm is the subject of the following subsection.

2.1.3. Self-Supervised Learning

Self-supervised learning is a learning paradigm whose central objective is to generate a supervisory signal automatically from the structure of unlabelled data, so that the standard machinery of supervised optimization can be applied without any manual annotation (Goodfellow et al., 2016; LeCun et al., 2015). It occupies a distinctive position between supervised and unsupervised learning. Like unsupervised learning, it requires no manually annotated targets, yet like supervised learning it formulates an explicit prediction objective with a well-defined loss function. The supervisory signal is typically obtained by withholding part of each input example from the model and asking it to predict the withheld portion from the remaining context, so that the data effectively labels itself.

Concrete instantiations of this idea have been proposed at several levels of representation. At the lexical level, word-embedding methods such as word2vec (Mikolov et al., 2013a,b) and fastText (Bojanowski et al., 2017) train shallow networks to predict a word from its neighbouring words, or vice versa, and use the learned hidden representation as a general-purpose word vector. At the level of full sentences, masked language modelling, also called discriminative, withholds a subset of tokens and trains the model to predict them from their surrounding context, such as the Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019), while generative language modelling withholds the rest of the sequence and trains the model to predict the next token given all preceding tokens, such as Generative Pre-trained Transformers (GPT) (Radford et al., 2019a). In computer vision, image-domain analogues include predicting the relative position of two image patches (Doersch et al., 2015) and reconstructing masked image regions (He et al., 2022). Once the model has been trained on such an objective, the resulting representations can be used directly, as in word embeddings, or transferred to downstream tasks by attaching a small task-specific head and fine-tuning on a comparatively small labelled dataset. The practical motivation for the paradigm is straightforward, as in most application domains, unlabelled data is abundant and inexpensive to collect, while manual annotation is costly, time-consuming, and in some cases requires specialized expertise.

One of the fields in which self-supervised learning has had a particularly strong impact is Natural Language Processing (NLP), where pre-trained language models have substantially raised the baseline of achievable performance across many tasks (Devlin et al., 2019). Two complementary self-supervised objectives are most common in this setting. The first, autoregressive language modelling, predicts each token given the tokens that precede it and underpins generative models such

as the GPT family (Radford et al., 2019a); this objective is discussed in more detail in Chapter 5. The second, Masked Language Modelling (MLM), is described here as a representative example of how an automatically generated supervisory signal can be used to train a deep encoder for representation learning. In MLM, a fraction of the input tokens in a text sequence are randomly replaced with a special mask token, and the model is trained to predict the identity of the masked tokens from their surrounding context. Formally, given an input sequence of tokens $\mathbf{x} = (x_1, x_2, \ldots, x_T)$ and a set of masked positions $\mathcal{M} \subset \{1, \ldots, T\}$, the masked language modelling objective is shown in Equation 1.

$$\mathcal{L}_{\text{MLM}} = \sum_{t \in \mathcal{M}} \log p(x_t \mid \mathbf{x}_{\setminus \mathcal{M}}; \boldsymbol{\theta}), \tag{1}$$

Here, $\mathbf{x}_{\setminus \mathcal{M}}$ denotes the input sequence with the masked positions replaced and $\boldsymbol{\theta}$ represents the model parameters (Devlin et al., 2019). The depth of the resulting contextual representations stems primarily from the underlying architecture rather than from the objective alone, since pairing MLM with a Transformer encoder, whose self-attention layers are bidirectional, allows the prediction of each masked token to draw on both left and right context. The concrete model families that result from this combination, in particular the BERT and the Robustly Optimized BERT Pretraining Approach (RoBERTa) (Liu et al., 2019a), are described in Chapter 5.

The broader workflow into which self-supervised pre-training fits, namely the pre-train then fine-tune paradigm, is discussed in detail in Chapter 5, and only the aspects directly relevant to the present discussion are noted here. The pre-trained weights act as an informed initialization that biases the optimization during fine-tuning toward solutions consistent with the linguistic patterns the model has already learned from the pre-training corpus and objective (Goodfellow et al., 2016). This is particularly valuable when the labelled dataset for the downstream task is small, since the pre-trained representations already encode much of the structure that the task-specific objective would otherwise have to learn from scratch. In this sense, self-supervised pre-training serves a regularisation-like function, encoding prior knowledge about the data domain into the model parameters before any task-specific labels are seen.

2.1.4. Loss Functions and Optimization

Training a machine learning model is, at its core, an optimization problem (LeCun et al., 2015; Jurafsky and Martin, 2026). The loss function $\mathcal{L}(\boldsymbol{\theta})$ is a scalar-valued function that quantifies how far a model's predictions are from the ground truth labels, given the current values of the model parameters $\boldsymbol{\theta}$ (Goodfellow et al., 2016; Bishop and Nasrabadi, 2006). The training process seeks to find the parameter

values that minimize this discrepancy over the training set, formally expressed in Equation 2.

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(f(\mathbf{x}_i;\, \boldsymbol{\theta}),\, y_i)\,, \tag{2}$$

Here, $f(\mathbf{x}_i;\, \boldsymbol{\theta})$ is the model's prediction for the i-th input and y_i is the corresponding ground truth label. Equation 2 is the empirical risk minimization objective, which is the guiding principle behind most supervised learning algorithms (Shalev-Shwartz and Ben-David, 2014). The choice of loss function is not arbitrary. It encodes assumptions about the distribution of the target variable and has a direct effect on the behaviour of the optimizer and the solutions it finds.

For classification tasks, the most widely used loss function is cross-entropy loss, which arises naturally from the principle of maximum likelihood estimation (Bishop and Nasrabadi, 2006; Goodfellow et al., 2016). In the binary case, the model outputs a single scalar $\hat{p} = f(\mathbf{x};\, \boldsymbol{\theta}) \in (0, 1)$, interpreted as the estimated probability of the positive class. Assuming the labels follow a Bernoulli distribution parameterized by $\hat{p}$, maximizing the log-likelihood over the training set is equivalent to minimizing the binary cross-entropy loss, as shown in Equation 3.

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{n} \sum_{i=1}^{n} \left[y_i \log \hat{p}_i + (1 - y_i) \log(1 - \hat{p}_i) \right], \tag{3}$$

In this equation, $y_i \in \{0, 1\}$ is the true label and $\hat{p}_i$ is the predicted probability for the i-th example (Bishop and Nasrabadi, 2006; Goodfellow et al., 2016). For multi-class problems with K classes, the model outputs a probability vector $\hat{\mathbf{p}} \in \mathbb{R}^K$ via a softmax activation, and the categorical cross-entropy loss generalizes as shown in Equation 4.

$$\mathcal{L}_{\text{CE}} = -\frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{K} y_{ik} \log \hat{p}_{ik}, \tag{4}$$

In this equation, y_{ik} is the one-hot encoded true label for class k and $\hat{p}_{ik}$ is the predicted probability assigned to class k for the i-th example (Bishop and Nasrabadi, 2006). Mean Squared Error (MSE) is another common loss function, but it is generally less suitable for classification. Because MSE penalizes errors quadratically and does not correspond to any natural probabilistic interpretation of a classification model's output, it tends to produce poorly calibrated probability estimates and slower, less stable training compared to the cross-entropy formulation given in Equation 4 (Goodfellow et al., 2016).

Once the loss function is defined, the model is trained by iteratively updating $\boldsymbol{\theta}$ in the direction that most reduces $\mathcal{L}$. Gradient descent is the foundational algorithm for this purpose. During gradient descent, at each step, the parameters are updated

by moving a small distance in the direction opposite to the gradient of the loss, presented in Equation 5 (Goodfellow et al., 2016).

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}), \tag{5}$$

In its basic form, the gradient in Equation 5 is computed over the entire training set, which is known as batch gradient descent. While this produces an exact gradient, it is computationally expensive and impractical for large datasets. In contrast, Stochastic Gradient Descent (SGD) estimates the gradient using a single randomly selected example at each step, which is cheap but produces noisy updates with high variance. The standard approach in practice is mini-batch gradient descent, which computes the gradient over a small randomly sampled subset of the training data at each update step. This balances computational efficiency with the quality of the gradient estimate, and the random element introduced by random mini-batch sampling has the additional benefit of helping the optimizer escape shallow local minima (Goodfellow et al., 2016). In a deep neural network, the gradient $\nabla_{\boldsymbol{\theta}} \mathcal{L}$ that these updates require is itself obtained through the backpropagation algorithm, which applies the chain rule of calculus to propagate the loss gradient backwards through the layers of the network (Rumelhart et al., 1986; Goodfellow et al., 2016). Its detailed formulation is presented later in Equation 7.

The learning rate η in Equation 5 applies a single, uniform step size to all parameters regardless of the curvature of the loss surface or the history of past gradients. Adaptive optimizers address this limitation by maintaining per-parameter learning rates that are adjusted based on the history of gradient magnitudes. The Adam optimizer (Kingma and Ba, 2015), which is used in this master's thesis, combines two ideas of momentum and adaptive scaling. Momentum accumulates a moving average of past gradients to smooth out updates. Adaptive scaling divides the update for each parameter by the square root of the moving average of past squared gradients. Concretely, given the first moment estimate $\mathbf{m}_t$, the second moment estimate $\mathbf{v}_t$, and a small constant ϵ for numerical stability, the Adam update at step t is given in Equation 6 (Kingma and Ba, 2015).

$$\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \eta \frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t} + \epsilon} \tag{6}$$

The effect of Equation 6 is that parameters associated with large or frequent gradient updates receive smaller effective step sizes, while parameters with small or infrequent gradients receive larger steps. This is particularly beneficial in deep networks where gradients can vary dramatically across layers and where sparse gradients are common, as is the case in models that process high-dimensional input representations (Goodfellow et al., 2016). A practical variant of this optimizer, AdamW, is used during fine-tuning of the proposed model in this master's thesis.

Loshchilov and Hutter (2019) showed that the L2 weight-decay penalty originally folded into the Adam update interacts unfavourably with adaptive scaling, and proposed decoupling weight decay from the gradient-based update so that it is applied as a separate parameter shrinkage step. The resulting AdamW optimizers update preserves the adaptive behaviour of Equation 6 while restoring the regularizing effect that weight decay is intended to provide, which has made it a common choice for fine-tuning large pre-trained Transformer models.

As introduced above, computing the gradient $\nabla_{\boldsymbol{\theta}}\mathcal{L}$ required in Equation 5 and Equation 6 for a deep neural network with many layers relies on the backpropagation algorithm (Rumelhart et al., 1986; Goodfellow et al., 2016; Bishop and Nasrabadi, 2006). Concretely, backpropagation propagates the gradient of the scalar loss backwards through the computational graph of the network, from the output layer to the input. For a composition of functions $\mathcal{L} = \ell \circ f^{(L)} \circ \cdots \circ f^{(1)}$, the gradient with respect to the parameters of layer l is computed as shown in Equation 7 (Goodfellow et al., 2016).

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}^{(l)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{(l)}} \cdot \frac{\partial \mathbf{a}^{(l)}}{\partial \boldsymbol{\theta}^{(l)}} \tag{7}$$

The key insight captured by Equation 7 is that intermediate gradient terms can be reused across layers, making the computation of the full gradient no more expensive than a single forward pass through the network. Without this efficiency, training deep networks with millions of parameters would be computationally impossible and very inefficient.

2.1.5. Evaluation Metrics for Classification

Before any individual metric can be computed, it is necessary to understand the structure of the errors a classifier can make. The confusion matrix provides exactly this measure. It is a complete collection of information about how a model's predictions relate to the true labels across all examples in the evaluation set (Bishop and Nasrabadi, 2006; Hastie et al., 2009). For a binary classifier, the confusion matrix is a 2×2 table whose cells are defined by four quantities. True Positives (TP) are examples that belong to the positive class and are correctly predicted as such. True Negatives (TN) are examples that belong to the negative class and are correctly predicted as negative. False Positives (FP), also known as Type I errors, are negative examples that are incorrectly assigned to the positive class. False Negatives (FN), also known as Type II errors, are positive examples that are incorrectly assigned to the negative class. In the context of the detection task in this master's thesis, a false negative corresponds to a failure to detect a true event of interest, while a false positive corresponds to a false alarm raised on a normal instance. These two error types carry different consequences, and the metrics

defined below in terms of the counts TP, FP, TN, and FN allow this asymmetry to be quantified explicitly.

When looking at the evaluation metrics, precision and recall are the two most informative threshold-dependent metrics for classification tasks where the cost of different error types is unequal. Precision is defined as the fraction of positive predictions that are actually correct, as shown in Equation 8.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{8}$$

Recall, also known as sensitivity or the true positive rate, measures the fraction of true positive examples that the model successfully identifies, as shown in Equation 9.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{9}$$

Precision and recall are not independent, because adjusting the decision threshold of a classifier creates a tradeoff between them (Hastie et al., 2009; Bishop and Nasrabadi, 2006). Lowering the threshold causes the model to predict the positive class more readily, which tends to increase recall at the cost of precision, since more false positives are accepted. Raising the threshold has the opposite effect. The F1 score summarizes this tradeoff in a single scalar by taking the harmonic mean of precision and recall, visible in Equation 10.

$$F_1 = 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{10}$$

The harmonic mean in Equation 10 is used rather than the arithmetic mean because it penalizes large imbalances between precision and recall. A classifier with very high precision but very low recall, or vice versa, will receive a low F1-score even if its arithmetic average appears moderate (Hastie et al., 2009). The F1-score is therefore a more demanding and informative summary than a simple average of the two quantities, particularly in settings where both error types matter.

Accuracy is perhaps the most intuitive classification metric, defined as the proportion of all examples that are correctly classified, as shown in Equation 11.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{11}$$

Another metric suitable for evaluating classifier performance is the Receiver Operating Characteristic (ROC) curve, which provides a threshold-independent view of classifier performance by plotting the True Positive Rate (TPR), which is identical to recall as defined in Equation 9, against the False Positive Rate (FPR)

across all possible decision thresholds (Hastie et al., 2009; Bishop and Nasrabadi, 2006). The FPR is defined in Equation 12.

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}} \tag{12}$$

A scalar summary of the ROC curve that is itself threshold-independent is the Area Under the Curve (AUC), written as AUC-ROC when computed for the ROC curve and defined in Equation 13.

$$\mathrm{AUC} = \int_0^1 \mathrm{TPR}(\mathrm{FPR})\, d\,\mathrm{FPR} \tag{13}$$

The quantity in Equation 13 admits a probabilistic interpretation, as it equals the probability that the model assigns a higher score to a uniformly drawn positive example than to a uniformly drawn negative example, so that an AUC of 0.5 corresponds to random ranking and an AUC of 1 corresponds to perfect separation (Fawcett, 2006; Bishop and Nasrabadi, 2006). Because the AUC integrates classifier behaviour across the full range of operating points, it is particularly informative in settings where the eventual decision threshold is unknown at training time, or where different deployment scenarios may impose different costs on the two error types.

2.1.6. Overfitting, Regularization, and Generalization

The ultimate goal of supervised learning is not to perform well on the training data, but to perform well on new, unseen data drawn from the same underlying distribution. This property is called generalization, and it is the central challenge of machine learning (Shalev-Shwartz and Ben-David, 2014; Goodfellow et al., 2016). Formally, the generalization gap is the difference between the empirical risk, defined as the average loss measured on the training set, and the expected risk, defined as the average loss one would observe over the full data-generating distribution. A model that achieves a low training loss, but a high expected loss has failed to generalize. This means the model has learned something specific to the training examples rather than something that reflects the true structure of the problem. Measuring and controlling this gap is therefore as important as minimizing the training loss itself.

The two canonical failure characteristics of supervised learning are overfitting and underfitting, and both can be understood through the lens of the bias-variance decomposition (Bishop and Nasrabadi, 2006; Hastie et al., 2009). For a regression setting, the expected squared error of a model $\hat{f}$ at a point $\mathbf{x}$ can be decomposed as seen in Equation 14.

$$\mathbb{E}\Big[(y - \hat{f}(\mathbf{x}))^2\Big] = \mathrm{Bias}^2\Big[\hat{f}(\mathbf{x})\Big] + \mathrm{Var}\Big[\hat{f}(\mathbf{x})\Big] + \sigma^2, \tag{14}$$

Here, σ^2 is the irreducible noise in the data. The bias term in Equation 14 captures the error introduced by approximating a complex target function with a model that is too simple, while the variance term captures the model's sensitivity to fluctuations in the training data (Hastie et al., 2009). Underfitting corresponds to high bias and low variance, meaning the model lacks the capacity to capture the underlying pattern and produces consistently poor predictions regardless of which training sample is used. Overfitting corresponds to low bias and high variance, meaning the model is expressive enough to fit the training data very closely, including its noise, but the learned function changes substantially with small changes in the training set and therefore fails to generalize. Deep neural networks sit firmly in the high-variance regime by default (Jurafsky and Martin, 2026), which is why explicit strategies to control overfitting are an essential part of their training.

One of the most direct ways to discourage overfitting is to add an explicit penalty term to the training objective that discourages the model from assigning large values to its parameters. This approach is known as regularization, and the two most common variants are L2 regularization and L1 regularization. In L2 regularization, also called weight decay, the penalty is proportional to the sum of squared parameter values (Bishop and Nasrabadi, 2006; Hastie et al., 2009). The goal is to shrink all weights smoothly toward zero during optimization, preventing any single parameter from growing so large that the model's predictions become dominated by a small subset of features. In L1 regularization, also known as the Lasso, the penalty is instead proportional to the sum of absolute parameter values. Because the L1 penalty has a non-differentiable point at zero, it tends to produce sparse solutions in which many parameters are driven to exactly zero, effectively performing implicit feature selection (Hastie et al., 2009). Both penalties have a natural Bayesian interpretation. L2 regularization corresponds to placing a zero-mean Gaussian prior over the parameters, while L1 regularization corresponds to a zero-mean Laplace prior (Bishop and Nasrabadi, 2006). Minimizing the regularized objective is then equivalent to finding the Maximum a Posteriori (MAP) estimate of the parameters under the respective prior.

Dropout is a regularization technique designed specifically for neural networks that operate in a fundamentally different way from the penalty-based methods described above (Goodfellow et al., 2016). During each forward pass in training, each unit in a given layer is independently set to zero with a fixed probability p, effectively removing it from the network for that update step. The remaining units are scaled by $1/(1-p)$ to preserve the expected magnitude of the layer's output. The practical effect is that the network can never rely on any single unit being present, which forces it to learn redundant, distributed representations that are more robust to noise in the input. A useful way to understand why dropout works is through the ensemble interpretation. Because a different subset of units is

deactivated at each training step, training with dropout is approximately equivalent to training an exponentially large number of different subnetworks and averaging their predictions at test time (Goodfellow et al., 2016). This implicit ensembling effect is a powerful source of regularization without requiring the explicit cost of training multiple separate models.

2.2. Deep Learning and Neural Networks

Deep learning refers to a class of ML methods built upon neural networks with multiple layers of computation, and it has fundamentally changed what is achievable in fields ranging from computer vision to speech recognition and natural language processing (LeCun et al., 2015; Goodfellow et al., 2016; Jurafsky and Martin, 2026). Rather than relying on hand-crafted features designed by domain experts, deep learning systems learn representations of data directly from raw inputs through a sequence of non-linear transformations, each layer building on the abstractions formed by the one before it. This capacity for automatic feature discovery is what distinguishes deep learning from earlier machine learning paradigms and explains much of its success across diverse and challenging tasks (Goodfellow et al., 2016; Jurafsky and Martin, 2026).

The theoretical foundations of deep learning rest on a combination of classical ideas from statistics, numerical optimization, and computational neuroscience, assembled into a framework that has become computationally possible only in recent decades through advances in hardware, software, and the availability of large labelled datasets (LeCun et al., 2015). Understanding how these pieces fit together is essential for interpreting the behaviour of any system built on deep learning principles, and it provides the knowledge needed to evaluate design choices with respect to both model capacity and generalization. This section develops that understanding systematically, beginning with the mathematical structure of neural networks and building toward the practical techniques that govern how such models are trained and deployed.

2.2.1. Neural Network Architectures

Neural networks are a family of parameterized function approximators that draw loose inspiration from the organization of biological nervous systems, though the connection to neuroscience is more motivational than literal (Goodfellow et al., 2016; Bishop and Nasrabadi, 2006; Jurafsky and Martin, 2026). The fundamental computational unit is the artificial neuron, which takes a vector of input values, computes a weighted sum of those inputs, adds a scalar bias term, and then passes the resulting value through an activation function. In general, an activation function

is any function applied to the pre-activation of a neuron in order to determine its output from the weighted combination of its inputs, and the function may in principle be linear or non-linear. In practice, however, the expressive power of a deep network depends crucially on the use of non-linear activation functions. Stacking many linear layers is mathematically equivalent to a single linear transformation and would collapse the hypothesis class of a deep model to that of a shallow one. It is therefore the introduction of non-linearity at each layer that allows neural networks to represent the rich, hierarchical structure that motivates their use in the first place. Formally, for an input vector $\mathbf{x} \in \mathbb{R}^n$, a weight vector $\mathbf{w} \in \mathbb{R}^n$, a bias $b \in \mathbb{R}$, and a non-linear activation function σ, the output of a single artificial neuron is given by the expression in Equation 15 (Goodfellow et al., 2016).

$$z = \sigma\big(\mathbf{w}^\top \mathbf{x} + b\big)\,, \tag{15}$$

Here, the quantity $\mathbf{w}^\top \mathbf{x} + b$ is referred to as the pre-activation (Goodfellow et al., 2016). The expression in Equation 15 is rather simple, and it is precisely this simplicity that makes the artificial neuron a foundational building block. The real power of neural networks does not reside in any individual unit but in the arrangement of many such units into layers and the composition of those layers into a deep hierarchy. When neurons are stacked in this way, the network as a whole can represent functions of considerable complexity, built up from the interactions of many elementary computations together (Bishop and Nasrabadi, 2006; Jurafsky and Martin, 2026). Each layer receives the outputs of the preceding layer as its inputs, transforms them, and passes the result forward, so that the final output of the network is the product of a long chain of such transformations applied in sequence (Goodfellow et al., 2016).

A central theoretical result that underpins the study of neural networks is the universal approximation theorem, which establishes that a feedforward network containing a single hidden layer of sufficient width, using a non-linear activation function, can approximate any continuous function defined on a compact subset of $\mathbb{R}^n$ to any desired degree of precision (Hornik, 1991; Goodfellow et al., 2016). This result guarantees that the class of functions representable by even a shallow neural network is, in theory, unlimited. However, the theorem is also importantly limited in what it does and does not say (Goodfellow et al., 2016). It says nothing about whether the approximating function can actually be found by gradient-based optimization given finite data, nor does it say how many hidden units would be required to achieve a given precision for a given target function. In the worst case, the number of units needed by a single hidden layer to represent a function that a deep network can represent compactly may grow exponentially with the input dimension (Goodfellow et al., 2016; Hornik, 1991). The theorem therefore establishes representational capacity as a theoretical ceiling but provides

no practical guarantee that this capacity is exploitable under realistic conditions. This limitation motivates the move away from shallow architectures toward deeper ones, where the composition of many layers can represent complex functions far more efficiently than any single layer of equivalent parameter count.

The practical advantages of depth over width have been confirmed repeatedly in both theoretical analysis and empirical work, and they are most naturally understood through the lens of hierarchical representation learning (LeCun et al., 2015; Goodfellow et al., 2016; Jurafsky and Martin, 2026). In a deep network, early layers tend to learn low-level features that are local and simple. As information propagates through deeper layers, these primitive features are combined and recombined into increasingly abstract representations. Intermediate layers capture part-level structure, while later layers encode semantically meaningful concepts that are invariant to a wide range of surface-level variations in the input (LeCun et al., 2015). It is important to mention that this explanation is simplified for the purpose of explaining the basic principle, but may not be true for all existing neural architectures. This hierarchical organization mirrors, at a high level, the structure of sensory processing systems observed in biological organisms, and it provides a principled account of why deep networks generalize well across tasks. The intermediate representations learned in one domain are often useful in related domains, because the underlying structure of the physical world is itself hierarchical in its nature (Goodfellow et al., 2016). The ability to discover and exploit this hierarchy automatically, without the need for hand-designed feature extractors, is one of the defining contributions of deep learning as a paradigm.

The broader field of deep learning encompasses several distinct families of neural network architectures, each designed around a different set of assumptions about the structure of the data and the nature of the task (LeCun et al., 2015; Goodfellow et al., 2016). Feedforward networks, also known as multilayer perceptrons (Haykin, 1994), are the most general form. They impose no structural constraints on the connectivity pattern between layers and are applicable to any fixed-dimensional input, making them a natural baseline and a common component within larger systems. Convolutional neural networks, whose modern form was introduced by LeCun et al. (1998), exploit the spatial or temporal locality and translation invariance of structured data such as images, audio signals, and time-series by replacing fully connected layers with local filtering operations and weight sharing, which drastically reduces the number of parameters while preserving sensitivity to local patterns regardless of their position in the input (LeCun et al., 1998, 2015). Recurrent neural networks and their gated variants, such as the Long Short-Term Memory (LSTM) network, are designed for sequential data in which the relevant context extends over time. They maintain a hidden state that is updated at each step of the sequence, allowing information from earlier time steps to influence predictions

at later ones (Goodfellow et al., 2016). More recently, attention-based architectures (Bahdanau et al., 2014), most prominently the Transformer (Vaswani et al., 2017), have emerged as a highly expressive alternative that models dependencies between all positions in a sequence simultaneously through a learned weighting mechanism, without the inductive bias of locality or sequentiality (Goodfellow et al., 2016; Jurafsky and Martin, 2026). Each of these families encodes a different inductive bias, and the appropriate choice depends on the structure of the problem at hand. The work presented in this master's thesis is built on Transformer-based architectures, motivated by their capacity to capture long-range contextual dependencies through the self-attention mechanism and their strong empirical performance on natural language understanding tasks (Vaswani et al., 2017; Devlin et al., 2019).

Among these families, feedforward networks are the most straightforward to describe in detail and provide the foundation on which more specialized architectures are built (Goodfellow et al., 2016; Bishop and Nasrabadi, 2006). A feedforward network, or multilayer perceptron, consists of a sequence of layers, each of which applies an affine transformation to its input followed by an element-wise non-linear activation function. The input layer holds the raw data unchanged, the hidden layers apply successive non-linear transformations that progressively reshape the input into increasingly abstract representations, and the output layer produces the network's prediction. The collection of all weight matrices and bias vectors across layers constitutes the learnable parameters of the network, and the forward pass simply applies these layer-wise transformations in sequence, propagating the input through the entire chain to produce a final output (Bishop and Nasrabadi, 2006). The expressive power of this construction grows with depth because each additional layer can learn a new non-linear remapping of the representation produced by all preceding layers, enabling the network to approximate functions of increasing complexity (Goodfellow et al., 2016).

To convert the output of a feedforward network into a prediction suitable for classification, the final layer typically applies a normalizing function to its raw scores. For multi-class problems, the softmax function maps a vector of real-valued scores to a valid probability distribution over the classes, while for binary classification the sigmoid function maps a single score to a probability in the interval $(0, 1)$ (Goodfellow et al., 2016). In either case, the network defines a conditional distribution over class labels given the input, and training proceeds by adjusting the parameters to maximize the likelihood of the observed labels under this distribution, which is equivalent to minimizing the cross-entropy loss (Bishop and Nasrabadi, 2006; Goodfellow et al., 2016).

Training proceeds by adjusting the parameters of the entire stack to minimize this loss, using the backpropagation procedure already described in Equation 7 (Rumelhart et al., 1986). The same per-layer chain-rule expansion applies regard-

less of the architectural family, so the optimization machinery introduced in the preceding subsections transfers directly to feedforward networks of arbitrary depth.

3. Transformer Architecture and Attention Mechanisms

This chapter details the Transformer architecture (Vaswani et al., 2017) and the attention mechanisms that form the backbone of many modern natural language processing systems. It begins with the fundamentals of attention, tracing the development from early sequence-to-sequence attention (Bahdanau et al., 2014) to the self-attention and cross-attention mechanisms surveyed in the broader attention literature (Chaudhari et al., 2021). It then examines the full Transformer architecture, including positional encodings, layer normalization, and feedforward sublayers, and compares encoder-only, decoder-only, and encoder-decoder variants. The chapter closes with a discussion of gating mechanisms for dynamic feature selection, which provide the conceptual foundation for the gated fusion approach proposed in this master's thesis.

3.1. Attention Mechanisms

Early sequence-to-sequence models relied on an encoder-decoder framework in which a recurrent encoder processed an input sequence of arbitrary length and compressed its contents into a single vector that had fixed dimensions. Afterwards, the decoder consumed this vector to produce the output sequence token by token (Sutskever et al., 2014; Cho et al., 2014). This design presented an information bottleneck, because all data relevant to the input, regardless of its length or complexity, had to be represented within that single context vector. In practice, this meant that as input sequences grew longer, the fixed-length representation became increasingly inadequate, and the quality degraded noticeably on longer sentences (Cho et al., 2014). The model was forced to choose, either explicitly or implicitly, which aspects of a long input to preserve and which to discard, and there was no mechanism to recover information that had been compressed away. This limitation may have provided practical motivation for the development of attention mechanisms, which wanted to give the decoder dynamic access to the full sequence of encoder representations rather than a single summarized vector.

The attention mechanism introduced by Bahdanau et al. (2014) resolved the fixed-length bottleneck by allowing the decoder to see all encoder hidden states at

every step of output generation. Rather than reading from a single compressed summary, the decoder computed a context vector at each decoding step as a weighted combination of encoder hidden states, where the weights reflected the relevance of each encoder state to the current decoding position. Formally, given the decoder hidden state $\mathbf{s}_{i-1}$ at step i and the j-th encoder hidden state $\mathbf{h}_j$, the alignment score is computed as shown in Equation 16,

$$e_{ij} = \mathbf{v}_a^\top \tanh(\mathbf{W}_a \, \mathbf{s}_{i-1} + \mathbf{U}_a \, \mathbf{h}_j) \tag{16}$$

where $\mathbf{W}_a$, $\mathbf{U}_a$, and $\mathbf{v}_a$ are learned parameters of a small feedforward network. Its role is to assess the compatibility between the decoder state and each encoder state. These raw scores are then converted into a normalized probability distribution over the encoder positions via a softmax function, as expressed in Equation 17,

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \tag{17}$$

where T_x is the length of the input sequence and α_{ij} is the attention weight assigned to encoder position j when decoding position i. The context vector for decoding step i is then formed as the weighted sum given in Equation 18,

$$\mathbf{c}_i = \sum_{j=1}^{T_x} \alpha_{ij} \, \mathbf{h}_j \tag{18}$$

so that $\mathbf{c}_i$ is a soft blend of all encoder states, with the blend concentrating on those positions most relevant to the current decoding step (Bahdanau et al., 2014). Because the alignment model in Equation 16 is a differentiable feedforward network, the entire system including the encoder, the alignment model, and the decoder can be trained end-to-end via backpropagation.

Luong et al. (2015) subsequently proposed a family of simpler scoring functions that aimed to replace the additive feedforward network with direct mathematical operations between the decoder and encoder states. The dot-product scoring function, given in Equation 19, computes alignment as the inner product of the two vectors,

$$e_{ij} = \mathbf{s}_i^\top \mathbf{h}_j, \tag{19}$$

while the general scoring function, shown in Equation 20, introduces a bilinear weight matrix $\mathbf{W}_a$ that adds a small number of learnable parameters without the cost of a full feedforward network.

$$e_{ij} = \mathbf{s}_i^\top \mathbf{W}_a \, \mathbf{h}_j. \tag{20}$$

Compared to the additive formulation in Equation 16, the dot-product variant in Equation 19 is considerably faster to compute and requires no additional parameters,

but it is sensitive to the magnitude of the input vectors. When the dimensionality d of the representations is large, the dot products grow in magnitude and push the softmax into regions of very small gradient, thus degrading learning. Vaswani et al. (2017) addressed this directly by introducing a scaling factor, replacing the raw dot product with a score proportional to $1/\sqrt{d}$, a scaled dot-product attention that became widely used in modern Transformer models. The three scoring functions together thus form a progression from expressive but computationally costly additive attention, through the more efficient bilinear form, to the parameter-free scaled dot product attention.

At a higher level of abstraction, attention can be seen as a differentiable soft addressing mechanism over a memory of encoder states (Bahdanau et al., 2014; Chaudhari et al., 2021). The encoder hidden states represent a memory bank, each entry storing a representation of a particular input position. The decoder state acts as a query that is compared against each memory entry to determine its relevance, and the resulting weighted sum retrieves a blended value from that memory. This framing connects attention to the broader idea of memory-augmented neural networks, in which a controller learns to read from and write to an external memory in a differentiable manner (Graves et al., 2014). A property that distinguishes attention from earlier fixed-readout mechanisms is that the information retrieved depends on the current query, meaning the same memory bank yields a different readout at every decoding step and for every input instance. This input-dependent dynamic routing stands in contrast to the static bottleneck vector it replaced, and it is this property that allows the decoder to selectively focus on whichever parts of the input are most important to generating each output token.

Although attention was originally developed in the context of Machine Translation (MT), it was quickly adopted across a wide range of tasks (Chaudhari et al., 2021). The Transformer architecture (Vaswani et al., 2017) took this idea further by replacing the recurrent structure of the encoder and decoder with layers of self-attention, that are discussed in the next section.

3.2. Self-Attention and Multi-Head Attention

The attention mechanism described in the preceding section operates between two distinct sequences, with the decoder querying the encoder's hidden states to retrieve relevant information at each generation step. This form of attention, often called cross-attention, draws on representations from one sequence to inform the processing of another. Self-attention generalizes this idea by allowing every position within a single sequence to attend to every other position in that same sequence (Vaswani et al., 2017). This approach allows the model to capture dependencies between any two positions directly, regardless of how far apart those positions are in

the sequence. In a recurrent network, by contrast, information must travel through a chain of sequential hidden state transitions to connect distant positions, and this sequential propagation both limits parallelism and risks attenuation of long-range signals (Chaudhari et al., 2021). Self-attention eliminates this limitation.

The mathematical foundation of self-attention is the query-key-value formulation introduced by Vaswani et al. (2017). Given an input sequence represented as a matrix $\mathbf{X} \in \mathbb{R}^{n \times d_{\text{model}}}$, where n is the sequence length and d_{model} is the model dimensionality, three separate linear projections are applied to produce query Q, key K, and value V matrices as shown in Equation 21,

$$\mathbf{Q} = \mathbf{X}\mathbf{W}^Q, \quad \mathbf{K} = \mathbf{X}\mathbf{W}^K, \quad \mathbf{V} = \mathbf{X}\mathbf{W}^V \tag{21}$$

where $\mathbf{W}^Q, \mathbf{W}^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$ and $\mathbf{W}^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ are learned weight matrices. The query vector encodes what a given position is looking for, the key vector encodes what a position has to offer, and the value vector encodes the content that will be retrieved if that position is selected. The compatibility between queries and keys is measured via scaled dot-product attention, defined in Equation 22,

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V} \tag{22}$$

where the factor $\sqrt{d_k}$ in the denominator is needed for stable training (Vaswani et al., 2017). Without this scaling, the dot products between queries and keys grow in magnitude as d_k increases, which drives the softmax into regions where its output is sharply peaked and its gradient becomes vanishingly small, impeding learning. Dividing by $\sqrt{d_k}$ keeps the pre-softmax values in an area where the gradients remain informative.

A single application of Equation 22 computes one set of attention weights and retrieves one weighted combination of value vectors. Multi-head attention extends this idea by running h independent attention functions in parallel, each operating in a distinct learned subspace of the representation (Vaswani et al., 2017). For each head i, separate projections $\mathbf{W}_i^Q$, $\mathbf{W}_i^K$, and $\mathbf{W}_i^V$ are applied to obtain head-specific query, key, and value matrices, attention is computed according to Equation 22, and the resulting head outputs are concatenated and projected through a final matrix $\mathbf{W}^O$, as formalized in Equation 23,

$$\begin{aligned} \text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) &= \text{Concat}(\text{head}_1, \ldots, \text{head}_h)\,\mathbf{W}^O, \\ \text{head}_i &= \text{Attention}(\mathbf{Q}\mathbf{W}_i^Q,\, \mathbf{K}\mathbf{W}_i^K,\, \mathbf{V}\mathbf{W}_i^V) \end{aligned} \tag{23}$$

By partitioning the representational capacity across multiple attention heads, the model is encouraged to learn qualitatively different patterns of attention simultaneously. Different heads tend to specialize. Some may capture local syntactic

relationships between adjacent tokens, while others may track long-range semantic dependencies or coreference links spread across the sequence. This diversity of attention patterns, each operating in its own subspace and contributing to a shared final projection, is widely regarded as a key source of the Transformer's expressiveness and its ability to model complex structured relationships (Vaswani et al., 2017).

Despite its strengths, the scaled dot-product formulation in Equation 22 carries a significant computational cost. Because every position attends to every other position, computing the full attention matrix requires $\mathcal{O}(n^2 d)$ operations and $\mathcal{O}(n^2)$ memory, where n is the sequence length and d is the dimensionality of the representations (Vaswani et al., 2017). This quadratic dependence on sequence length is manageable for moderate sequence lengths, but may become problematic when processing very long documents, high-resolution images, or other inputs with many input tokens. A body of research has consequently explored strategies to reduce this cost, including sparse attention patterns that restrict each position to attending only to a local neighbourhood or a set of global tokens, low-rank approximations of the attention matrix, and linear attention variants that reformulate the computation to avoid materializing the full $n \times n$ matrix (Tay et al., 2022).

3.3. Transformer Architecture Components

The Transformer, as introduced by Vaswani et al. (2017), is organized around a single repeated structural unit called the Transformer block, sometimes also referred to as a Transformer layer. Each block contains two principal sublayers, that are a multi-head self-attention sublayer of the kind described in the preceding section, followed by a position-wise feedforward sublayer. Both sublayers are wrapped with a residual connection and a layer normalization operation, and the full encoder or decoder is constructed by stacking N such blocks in a continuous sequence. The uniformity of this design is one of its most consequential properties, as depth is achieved entirely through repetition of the same modular structure rather than by introducing specialized layer types at different stages. This means that increasing the capacity of the model reduces to a question of how many blocks to stack and how wide each block should be, decisions that can be made independently of the internal mechanics of any individual block (Vaswani et al., 2017).

A residual connection, also called a skip connection, is applied around each sublayer so that the input to a sublayer is added directly to its output before the result is passed forward. Formally, the output of a residual-wrapped sublayer is given in Equation 24,

$$\mathbf{y} = \mathbf{x} + \text{Sublayer}(\mathbf{x}) \tag{24}$$

where $\mathbf{x}$ is the sublayer input and $\text{Sublayer}(\cdot)$ denotes either the self-attention

function or the feedforward network. The effect of Equation 24 is that the sublayer is not required to learn the full desired transformation but only the residual correction to its input, which is a substantially easier optimization target. This formulation was introduced in the context of deep convolutional networks by He et al. (2016), who showed that residual connections alleviate the vanishing gradient problem by providing a direct path along which gradients can propagate back through many layers without passing through repeated non-linearities. In the Transformer, where models may span dozens or more of stacked blocks, these connections are also important, because without them, training deep Transformer stacks could be considerably less stable and could require much more careful initialization (Vaswani et al., 2017; He et al., 2016).

Each residual connection in Equation 24 is followed by a layer normalization operation, which standardizes the representation before it enters the next component. Layer normalization, proposed by Ba et al. (2016), computes the mean and variance of the activations across the feature dimension for each individual example independently, as expressed in Equation 25,

$$\mathrm{LayerNorm}(\mathbf{x}) = \frac{\mathbf{x} - \mu}{\sigma + \epsilon} \odot \boldsymbol{\gamma} + \boldsymbol{\beta} \tag{25}$$

where μ and σ are the mean and standard deviation computed over the feature dimension of $\mathbf{x}$, ϵ is a small constant for numerical stability, and $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ are learned scale and shift parameters. This stands in contrast to batch normalization, which aims to normalize across the batch dimension and therefore depends on statistics that vary with batch size and composition (Vaswani et al., 2017). Layer normalization is particularly well suited to the Transformer setting because it operates on each example independently, making it agnostic to batch size and naturally compatible with variable-length sequences. It is also compatible with autoregressive generation, where at inference time each token is produced one at a time and batch statistics would be undefined or meaningless. In the original formulation of Vaswani et al. (2017), layer normalization is applied after the residual addition, a placement known as post-norm.

The second sublayer within each Transformer block is the position-wise feedforward network, which applies the same two-layer fully connected transformation independently and identically to the representation at every sequence position. Its output is defined in Equation 26,

$$\mathrm{FFN}(\mathbf{x}) = \mathbf{W}_2\, \sigma(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2 \tag{26}$$

where σ denotes a non-linear activation function, originally the rectified linear unit by Vaswani et al. (2017), and the inner dimensionality of the first linear transformation is typically set to four times the model dimension d_{model}, producing

an expansion followed by a contraction back to d_{model}. In modern Transformer-based models, the ReLU activation may often be replaced by the Gaussian Error Linear Unit (GELU), a smooth approximation that weights the input by the standard Gaussian cumulative distribution function rather than by a hard zero threshold. Devlin et al. (2019) adopted GELU in BERT and subsequent encoder-only models such as RoBERTa kept it as the default, motivated by its smoothness near the origin and its slightly improved empirical optimization behaviour compared to ReLU. The hybrid architecture proposed in this master's thesis follows the same convention in its feedforward projection layers. Because the transformation in Equation 26 is applied separately at each position with no interaction across positions, the feedforward sublayer performs no cross-position mixing. That role belongs entirely to the self-attention sublayer. Instead, the feedforward sublayer provides each position with a rich per-position transformation that allows the network to perform complex non-linear feature processing on the representation produced by attention (Vaswani et al., 2017).

In retrospect, the complete information flow through a single Transformer block can be traced as follows. The block receives an input matrix $\mathbf{X}$ in which each row represents the current state of one sequence position. Multi-head self-attention, as defined in Equation 23, is applied to $\mathbf{X}$, allowing every position to gather context from every other position. The attention output is added back to $\mathbf{X}$ via the residual connection in Equation 24 and then normalized according to Equation 25, getting an intermediate representation in which each position has been enriched with global contextual information while the original signal remains accessible through the shortcut path. This intermediate representation is then passed through the position-wise feedforward network of Equation 26, which refines each position independently, and the result is again combined with the input to that sublayer via a second residual connection and normalized by a second layer normalization. This process ensures that the self-attention handles communication across positions, aggregating information from the full context, while the feedforward sublayer handles transformation within each position, deepening the representational abstraction. Residual connections and layer normalization together ensure that the signal remains well conditioned as it passes through many such blocks in sequence, enabling the representational hierarchy that deep networks rely on (Vaswani et al., 2017).

3.4. Positional Encodings and Input Representations

Self-attention, as defined in Equation 22, computes its output as a weighted sum of value vectors based on the similarity between query and key vectors. Since this operation treats all positions equally without regard to order, it cannot distinguish between different sets of the same tokens. For example, a Transformer without positional information would treat *the cat sat on the mat* and *mat the on sat cat the* the same way, despite their very different meanings. Because sequence order is important in language, time series, and images, the model must have explicit position information to work correctly (Vaswani et al., 2017).

The original Transformer of Vaswani et al. (2017) addressed order problem through sinusoidal positional encodings, in which a deterministic, parameter-free vector is constructed for each position and added element-wise to the corresponding token embedding before the sequence enters the first Transformer block. The encoding for position *pos* along embedding dimension i is defined by the pair of functions given in Equation 27 and Equation 28,

$$\mathrm{PE}(pos,\, 2i) = \sin\Big(\frac{pos}{10000^{2i/d_{\text{model}}}}\Big) \tag{27}$$

$$\mathrm{PE}(pos,\, 2i+1) = \cos\Big(\frac{pos}{10000^{2i/d_{\text{model}}}}\Big) \tag{28}$$

where d_{model} is the model dimensionality and i indexes the dimension pair. The frequencies decrease geometrically with dimension index, so that low-frequency sinusoids capture coarse positional distinctions while high-frequency sinusoids resolve fine-grained ones, producing a unique encoding pattern for each position. A useful property of this scheme is that the encoding at position $pos + k$ can be expressed as a fixed linear transformation of the encoding at position *pos*, which means the model can, in principle, learn to attend to relative positions by composing linear operations on the absolute encodings (Vaswani et al., 2017). Because the functions in Equation 27 and Equation 28 are deterministic, sinusoidal encodings introduce no additional parameters and can, in principle, generalize to sequence lengths longer than any encountered during training.

An alternative that has been widely accepted as the default choice in many modern Transformer-based models is the learned positional embedding, in which a distinct embedding vector is associated with each integer position index and optimized alongside all other model parameters during training (Vaswani et al., 2017; Devlin et al., 2019). The positional embedding for position *pos* is simply the *pos*-th row of a learned matrix $\mathbf{E}_{\text{pos}} \in \mathbb{R}^{L_{\max} \times d_{\text{model}}}$, where $L_{\max}$ is the maximum sequence length supported by the model. This vector is added to the token embedding in exactly the same way as a sinusoidal encoding, so the two schemes

are interchangeable from the perspective of downstream components. A limitation of learned embeddings is that the model cannot process sequences longer than $L_{\max}$, because no embedding exists for positions beyond that index. Vaswani et al. (2017) found that learned and sinusoidal encodings produced nearly identical translation quality, and subsequent large-scale pre-trained models such as BERT (Devlin et al., 2019) adopted learned positional embeddings as a practical default, a convention that has largely persisted in encoder-only and decoder-only architectures.

Both sinusoidal and learned positional encodings assign each token an absolute position index, but an alternative family of methods encodes the relative distance between pairs of positions rather than their absolute locations in the sequence. Shaw et al. (2018) proposed adding learnable scalar biases to the attention logits before the softmax, where the bias applied to the pair (i, j) depends only on the offset $i - j$ clipped to a maximum distance. This modification allows the attention function to directly model relational structure between positions, and because the biases depend only on relative offsets, they generalize naturally to sequence lengths not seen during training.

A more recent variant of positional encodings is the Rotary Position Embedding approach proposed by Su et al. (2024), which encodes relative positions by applying a rotation matrix to the query and key vectors before the dot product is computed, so that the inner product between a query at position i and a key at position j depends on the difference $i - j$. Rotary embeddings have the attractive property that they introduce no additional parameters and integrate directly into the scaled dot-product attention computation without modifying the attention formula itself.

In retrospect, the full input representation pipeline of a Transformer begins with a vocabulary of discrete tokens, each mapped to a dense vector through a learned embedding matrix $\mathbf{E}_{\text{tok}} \in \mathbb{R}^{V \times d_{\text{model}}}$, where V is the vocabulary size. The positional encoding or embedding for each position is then added to the corresponding token embedding, yielding a sequence of d_{model}-dimensional vectors that serve as the input to the first Transformer block. In some architectures, additional structured embeddings are superimposed at this stage; Devlin et al. (2019), for instance, added segment embeddings to distinguish the two sentences in a sentence-pair input, combining token, positional, and segment information into a single summed representation. The embedding dimensionality d_{model} is a global hyperparameter that propagates through every sublayer of every block, tying together the input representation and the internal dimensionality of the model. In many language models, the token embedding matrix $\mathbf{E}_{\text{tok}}$ is additionally tied to the output projection matrix that converts the final hidden state back into a distribution over the vocabulary, halving the parameter count associated with the embedding layer (Vaswani et al., 2017; Devlin et al., 2019).

3.5. Encoder-Only, Decoder-Only, and Encoder-Decoder Variants

The Transformer architecture proposed by Vaswani et al. (2017) was designed as a full encoder-decoder model for sequence-to-sequence machine translation. The encoder is a stack of N Transformer blocks in which every position attends to every other position without any masking, so each encoder representation is characterized by the full bidirectional context of the input sequence. In contrast, the decoder is a corresponding stack of N blocks that generates the output sequence one token at a time in an autoregressive fashion, meaning each generated token is conditioned on all previously generated tokens. To enforce this causal constraint, the self-attention sublayer within each decoder block applies a triangular mask that prevents any position from attending to future positions. After its masked self-attention sublayer, each decoder block contains a cross-attention sublayer in which the queries are derived from the decoder's own hidden states while the keys and values are derived from the encoder's output representations. This cross-attention sublayer is the connective tissue between the two stacks, and it performs precisely the same role as the attention mechanism described in the preceding section on attention mechanisms, allowing the decoder to selectively retrieve information from the encoded input at every generation step (Vaswani et al., 2017).

Encoder-only Transformers retain the encoder stack and discard the decoder entirely, relying on the bidirectional self-attention of the encoder to build rich contextual representations of the input. A canonical example of this model family is BERT, introduced by Devlin et al. (2019), which is pretrained on a masked language modelling objective in which a random subset of input tokens is replaced by a special mask token and the model is trained to predict the original tokens from their surrounding bidirectional context. Because the self-attention in an encoder-only model is not causally masked, every position can attend to every other position simultaneously, and the resulting representations encode both left and right context for each token. This makes encoder-only models well suited for tasks that require full understanding of the input rather than sequential generation. Examples of such tasks are text classification, named entity recognition, and dense retrieval. For sequence-level prediction tasks such as sentiment classification, Devlin et al. (2019) introduced the convention of prepending a special classification token, denoted `[CLS]`, to the input and using the final hidden state at that position as a summary representation of the entire sequence, which is then passed to a task-specific output layer.

Decoder-only models possess the masked self-attention of the decoder stack while removing both the encoder and the cross-attention sublayer, leaving a model that processes and generates a single unified sequence from left to right. The GPT series

of models (Radford et al., 2018, 2019a) established this architecture as a highly effective framework for language modelling, in which the training objective is to predict each token given all preceding tokens. The causal masking ensures that the prediction at each position depends only on the left context, which can be seen as a natural inductive bias for generation tasks and also makes the training signal dense, since every position in every sequence contributes a prediction loss. A defining characteristic of this architecture family is its scaling behaviour. Decoder-only models have been remarkably successful in performance improvements through increases in parameter count, dataset size, and compute budget, and they form the basis of the largest and most capable state-of-the-art LLMs available today (Radford et al., 2019a). Furthermore, decoder-only models can handle tasks that might seem to require an encoder-decoder structure, such as translation or summarization, by reformulating them as conditional text generation in which the input is prepended to the output as a prompt, a technique that has become central to the practical deployment of LLMs.

In retrospect, the three architectural variants differ along several interconnected dimensions that together determine their inductive biases and their natural domains of applicability (Lin et al., 2022; Raffel et al., 2020). Encoder-only models use fully bidirectional attention and are pretrained with a masked language modelling objective, which encourages deep contextual understanding of the input at the cost of not being natively generative. Decoder-only models use strictly causal attention and are trained with a next-token prediction objective, making them natural generators but limiting their per-token representations to left context only. Encoder-decoder models combine both attention approaches, using bidirectional encoding of the input and causal decoding of the output connected through cross-attention, which makes them well suited to tasks with a clear asymmetry between an input to be understood and an output to be generated, such as translation, summarization, and structured prediction (Vaswani et al., 2017; Raffel et al., 2020).

3.6. Gating Mechanisms and Dynamic Feature Selection

Gating is a general principle by which a given neural network learns to modulate, i.e., control the flow of information in a data-dependent manner rather than passing all signals forward uniformly. The idea was established in the context of recurrent networks through the LSTM architecture of Hochreiter and Schmidhuber (1997), in which dedicated input, forget, and output gates control what information is written into the cell state, what is discarded, and what is exposed to the next layer. The fundamental operation underlying all gating mechanisms is a multiplicative

interaction between a signal and a learned gate vector whose values are bounded between zero and one, typically by a non-linear sigmoid activation function. By element-wise multiplying a gate vector with a feature vector, the network can selectively amplify or suppress individual dimensions of the representation based on the current input, giving it a flexible and differentiable form of dynamic feature selection (Hochreiter and Schmidhuber, 1997; Chaudhari et al., 2021).

Gated linear units, introduced by Dauphin et al. (2017), apply this gating principle within feedforward layers as a direct replacement for standard activation functions. In the original formulation, the input is projected into two separate linear transformations, one of which passes through a sigmoid to produce a gate, and the result is the element-wise product of the two branches, as expressed in Equation 29,

$$\mathrm{GLU}(\mathbf{x}) = (\mathbf{x}\mathbf{W}_1 + \mathbf{b}_1) \odot \sigma(\mathbf{x}\mathbf{W}_2 + \mathbf{b}_2)\,, \tag{29}$$

where σ denotes the sigmoid function and $\odot$ denotes element-wise multiplication. Subsequent work by Shazeer (2020) explored replacing the sigmoid in the gating branch with smoother non-linearities such as GELU or SiLU, getting variants that consistently outperform the standard ReLU feedforward sublayer in Transformer models across a range of language modelling benchmarks.

A complementary form of gating operates at the level of feature channels rather than individual neuron activations. The squeeze-and-excitation network of Hu et al. (2018) introduced a channel-wise gating block that can be inserted into any convolutional neural architecture. The block first compresses spatial information into a single descriptor per channel through global average pooling, as shown in Equation 30,

$$z_c = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} u_c(i, j), \tag{30}$$

where $u_c(i, j)$ is the activation of channel c at spatial location (i, j). The resulting channel descriptor $\mathbf{z}$ is then passed through a small two-layer bottleneck network with a sigmoid output to produce per-channel gate values, which are used to rescale the original feature map channels as given in Equation 31,

$$\tilde{\mathbf{u}}_c = s_c \cdot \mathbf{u}_c, \quad \mathbf{s} = \sigma(\mathbf{W}_2\, \delta(\mathbf{W}_1 \mathbf{z}))\,, \tag{31}$$

where δ is a ReLU activation and $\mathbf{W}_1$, $\mathbf{W}_2$ are the bottleneck weights. The effect of Equation 31 is that the network learns, for each input, which feature channels are most informative and up-weights them while suppressing less relevant ones, performing dynamic feature selection at a coarser granularity than the element-wise gating of Equation 29.

The same principle adapts naturally to the setting in which the model receives a single fixed-length vector of scalar engineered features rather than a stack of

convolutional feature maps. When there is no spatial axis to pool over, the squeeze step of Equation 30 becomes trivial, since each feature is already represented by a single scalar, and the excitation step reduces to learning a per-feature gate directly from the input feature vector itself. Formally, given a feature vector $\mathbf{f} \in \mathbb{R}^D$, a learned attention vector $\mathbf{a} \in (0,1)^D$ is produced from $\mathbf{f}$ through a small bottleneck network with a sigmoid output, and the re-weighted feature vector $\tilde{\mathbf{f}}$ is obtained as the element-wise product, as shown in Equation 32.

$$\tilde{\mathbf{f}} = \mathbf{a} \odot \mathbf{f}, \quad \mathbf{a} = \sigma(\mathbf{W}_2\, \delta(\mathbf{W}_1 \mathbf{f})) \tag{32}$$

The approach in Equation 32 equals to a squeeze-and-excitation block applied to a feature vector of unit spatial extent, and it inherits the same interpretation as Equation 31. Each scalar feature is treated as its own channel, and the network learns which of these channels carry the strongest discriminative signal for the current input. In the hybrid detection architecture proposed in this master's thesis, this mechanism is applied to a fixed-length vector of pre-extracted stylometric features before that vector is projected to the dimensionality of the Transformer representation, so that the model can choose whichever lexical, syntactic, discourse, or probability-based features are most informative for each individual passage rather than relying on a uniform weighting fixed at training time.

When features must be combined from multiple sources, such as different spatial scales, processing branches, or data modalities, simple addition or concatenation treats all sources as equally reliable regardless of the input. A gated fusion mechanism addresses this by learning input-dependent mixing proportions, as expressed in the general form given in Equation 33,

$$\mathbf{f}_{\text{fused}} = \mathbf{g} \odot \mathbf{f}_A + (1 - \mathbf{g}) \odot \mathbf{f}_B, \quad \mathbf{g} = \sigma(\mathbf{W}_g[\mathbf{f}_A;\, \mathbf{f}_B] + \mathbf{b}_g)\,, \tag{33}$$

where $\mathbf{f}_A$ and $\mathbf{f}_B$ are the two feature vectors to be fused, $[\cdot\,;\cdot]$ denotes concatenation, and $\mathbf{g}$ is a learned gate that determines the contribution of each source at every feature dimension (Chaudhari et al., 2021; Hu et al., 2018). This formulation is more expressive than a fixed weighted sum because the proportions in Equation 33 change with every input instance.

4. Natural Language Processing and Linguistic Features

This chapter introduces the Natural Language Processing (NLP) concepts and linguistic features used to analyse text computationally, drawing on the standard treatment of these topics by Jurafsky and Martin (2026) and Schaaff et al. (2023). It opens with the input-side machinery shared by all modern NLP systems, covering text representation and tokenization and the distributional view of meaning encoded in word embeddings. It then looks at the syntactic and semantic processing steps of the NLP pipeline, from Part-Of-Speech (POS) tagging and dependency parsing to discourse-level coherence modelling. The chapter then describes the families of linguistic features that quantify text at different levels of organization, including lexical and vocabulary statistics, syntactic complexity measures derived from dependency structure, discourse and coherence indicators, stylometric markers of writing style following the feature taxonomy of Schaaff et al. (2023), and LM-based metrics such as perplexity.

4.1. Text Representation and Tokenization

Neural networks operate exclusively on numerical inputs, yet human natural language is symbolic and discrete. Before any model can process a sentence, that sentence must be converted into a sequence of numerical values that preserve enough of the original linguistic structure to be useful for downstream tasks. This conversion is not a single operation but a mini-pipeline of decisions, beginning with how to segment raw text into meaningful units, and ending with how to represent those units as vectors in a continuous vector space (Jurafsky and Martin, 2026). The choices made at each stage of this pipeline have consequences that influence the entire model, including vocabulary coverage, computational cost, and the model's ability to generalize across languages and domains. Understanding these choices is therefore a necessary prerequisite for interpreting the behaviour of any Transformer-based system.

The most intuitive approach to text segmentation may be to split on whitespace and punctuation boundaries, treating each word as a unit of representation. This word-level tokenization is straightforward to implement and produces tokens that

correspond directly to human-readable lexical items. However, it carries a fundamental weakness in the form of a fixed, closed vocabulary. Any word that was not observed during training becomes an out-of-vocabulary token, losing all semantic information in a single step. Character-level tokenization sits at the opposite extreme, decomposing text into individual characters and thereby guaranteeing a tiny, complete vocabulary with no out-of-vocabulary problem. The cost of this completeness is that the model must learn to compose meaning from very short, individually uninformative units, which may be quite difficult for a model and tends to produce much longer sequences for the same input (Jurafsky and Martin, 2026). Neither extreme is satisfactory for large-scale language modelling, and practical systems have converged on strategies that interpolate between the two.

Subword tokenization has become the standard approach in modern natural language processing precisely because of a more effective trade-off between vocabulary size and linguistic coverage. Byte-pair encoding, introduced for neural machine translation by Sennrich et al. (2016), begins with a character-level vocabulary and iteratively merges the most frequent adjacent symbol pair until a target vocabulary size is reached. The result is a vocabulary in which common words appear as single tokens while rare or morphologically complex words are decomposed into familiar subword fragments. This decomposition allows the model to handle unseen words at inference time by falling back on the known fragments rather than resorting to a generic unknown symbol. WordPiece, the tokenization scheme used in BERT (Devlin et al., 2019), follows a closely related principle but selects merge candidates by maximizing the likelihood of the training corpus under a unigram language model rather than by raw frequency. Both methods produce vocabularies of a few tens of thousands of tokens, which is large enough to represent most common words as single units yet small enough to keep the embedding matrix computationally feasible.

4.2. Word Embeddings and Distributional Semantics

The distributional hypothesis, i.e., the observation that words appearing in similar contexts tend to have similar meanings, forms the theoretical foundation for representing words as dense real-valued vectors (Jurafsky and Martin, 2026). Under this framework, the shape of the resulting vector space encodes semantic relationships so that words that behave similarly in text occupy nearby positions. Two methods that adopted this idea at scale are Word2Vec (Mikolov et al., 2013b), which learns vectors by training a shallow neural network to predict a word from its neighbours or its neighbours from the word, and GloVe (Pennington et al., 2014), which factorizes

a global word co-occurrence matrix. Both produce compact, fixed-dimensional representations that transfer well to downstream tasks despite having been trained in a self-supervised manner.

Static embeddings assign a single vector to each word type regardless of how the word is used in a particular sentence, which means they cannot distinguish between the different senses of polysemous words or capture how meaning shifts with grammatical role and surrounding context. Transformer-based models such as BERT (Devlin et al., 2019) overcome this by computing a representation for each token that is conditioned on the entire input sequence through the self-attention mechanism described in Chapter 3.

4.3. Lexical Features and Vocabulary Statistics

Beneath the syntactic and semantic structure of a text lies a layer of surface-level vocabulary choices that reflect the habits, preferences, and limitations of its author. Lexical features attempt to quantify these patterns in ways that are both interpretable and computationally cheap to extract (Jurafsky and Martin, 2026). One of the most fundamental measures for this is the Type-Token Ratio (TTR), defined as the number of unique words divided by the total number of word tokens in a passage. A high ratio indicates a varied vocabulary, while a low ratio suggests repetition. Thus, both extremes carry stylistic implications that vary with author, genre, or generation method.

Function words, the closed-class items such as prepositions, conjunctions, and auxiliary verbs are also particularly informative for characterizing writing style because their selection is largely automatic and therefore may be resistant to deliberate manipulation (Schaaff et al., 2023). Their relative frequencies form a signature that has long been exploited in authorship attribution research. Complementing this, content word statistics such as average word length, the proportion of words exceeding a given syllable count, and the rate of hapax legomena, i.e., words appearing exactly once, capture vocabulary quality and density in ways that may differ systematically between human writers and text generated by LLMs.

Neural language models trained on massive corpora tend to produce text that is lexically plausible at the token level, but may differ from human writing in its distributional vocabulary patterns at the passage level (Schaaff et al., 2023). Lexical features, being computed over entire passages rather than individual token predictions, are sensitive to exactly these aggregate patterns and therefore may provide a signal that is not redundant with the contextual embeddings used in the neural branch of the architecture. This complementarity is a key motivation for including vocabulary statistics in the hybrid feature set described in Chapter 8.

4.4. Syntactic Features and Dependency Parsing

Syntactic analysis begins with assigning each token a grammatical category, a process known as POS tagging, which labels tokens as nouns, verbs, adjectives, adverbs, etc. according to their role in the sentence (Jurafsky and Martin, 2026). Modern taggers may achieve near-human accuracy on standard corpora by exploiting contextual information, and their output is the starting point for all higher-level syntactic processing. The distribution of POS tags across a text may serve as a coarse syntactic fingerprint, as a passage dense in nominalizations and adjectives exhibits a different structural profile from one relying heavily on finite verbs and adverbs. In addition, these profiles vary in characteristic ways between writing styles, as well as between human and LLM-generated text.

Beyond category labels, dependency parsing recovers the grammatical relations between tokens by constructing a directed graph in which each token except the root is linked to its syntactic head by a labelled arc, indicating the relation type, such as subject, direct object, or adverbial modifier (Nivre, 2008). This representation makes explicit the predicate-argument structure of each clause and allows the extraction of features that go beyond simple word order statistics. Transition-based and graph-based parsing algorithms have reached high accuracy on broad-coverage treebanks, making dependency parse trees a reliable and scalable source of structural information across diverse text genres.

A range of structural statistics can be computed from a dependency graph, which quantify sentence complexity in interpretable terms. Relevant features include, for instance, average dependency arc length, which measures how far each word is from its syntactic head and correlates with structural embedding and centre of gravity, parse tree depth, which reflects the degree of recursive embedding in the phrase structure, and the frequency of particular construction types such as passive clauses, relative clauses, and coordinated structures. These measures capture aspects of syntactic complexity that may differ systematically between human prose and the output of LLMs, which tend to favour shallower and more locally-predictable text structures (Schaaff et al., 2023).

4.5. Discourse and Coherence Features

While lexical and syntactic features characterize individual sentences, discourse-level analysis examines how sentences connect to form a coherent text. A coherent text is one in which the reader can follow a consistent thread of entities, events, and argumentative steps from beginning to end, whereas an incoherent one, even if each sentence is grammatical, leaves the reader unable to integrate successive propositions into a unified mental model (Jurafsky and Martin, 2026). Modelling

this property computationally requires stepping beyond the sentence boundary and considering the structural relationships that hold across the entire text or its parts.

One influential formal approach to coherence is the entity grid model (Barzilay and Lapata, 2008), which represents a text as a matrix recording the grammatical role, subject, object, or other, played by each discourse entity in each sentence. Transitions between roles across adjacent sentences encode how smoothly an entity is tracked through the text, and the distribution of these transitions serves as a quantitative coherence score. Complementing this structural view, discourse connectives, explicit lexical markers such as *however*, *therefore*, and *consequently*, signal the rhetorical relations holding between clauses and may provide a shallow but reliable indicator of argumentative organization.

LLMs generate text by predicting each token conditioned on the local context, without an explicit global planning mechanism that ensures discourse-level consistency. This may produce text that is locally fluent but globally incoherent in subtle ways, for instance repeating information, losing track of an introduced entity, or shifting topic without an appropriate connective signal (Schaaff et al., 2023). Coherence features that measure entity transition patterns and connective usage are therefore potentially discriminative for the detection task, capturing a dimension of structure that perplexity-based scores and surface statistics may miss.

4.6. Stylometric Features

Stylometry is the quantitative study of writing style, concerned with identifying patterns in an author's linguistic choices that persist across different topics and genres and that can be measured reliably from text alone (Stamatatos, 2009). In computational form, stylometric methods construct feature representations of texts that are sensitive to authorial style while remaining robust to topic variation, making them a long-standing source of features for any classification task that depends on the linguistic signature of a passage rather than on its semantic content.

The features employed in stylometric systems typically span several levels of linguistic organization (Stamatatos, 2009; Schaaff et al., 2023). Character n-gram frequencies capture low-level writing habits, including punctuation preferences and affixation patterns. Function word profiles, as discussed in the preceding section, encode grammatical style independently of content. Syntactic features such as those derived from POS tag sequences and parse tree statistics encode structural preferences at the sentence level. The joint distribution over these feature families provides a stylistic fingerprint that is more stable than any single feature class alone, because changing a passage's stylistic signature consistently across every level of organization is considerably harder than altering any one of these levels in isolation (Stamatatos, 2009).

4.7. Language Model-Based Features

A natural source of features for LLM-generated text detection may be another pretrained language model itself, used not as a generator but as a probabilistic scorer. Given a candidate passage, a reference model assigns a probability to each token conditioned on its left context, and the sequence of these conditional probabilities can be aggregated into passage-level statistics. Commonly extracted quantities include the mean log-probability per token, the token-level entropy of the model's predictive distribution, and the rank of the observed token among the full vocabulary sorted by probability (Jurafsky and Martin, 2026). Each of these features captures a different aspect of how expected or surprising the passage is under the model's learned distribution.

Text generated by a language model is, by construction, sampled from regions of high probability under that model, because decoding strategies such as nucleus sampling and top-k sampling explicitly constrain generation to the most probable tokens. Human writing, by contrast, is produced by cognitive processes unconstrained by any particular probability model and therefore occupies a much wider and less predictable region of text space (Schaaff et al., 2023). As a consequence, LLM-generated passages tend to exhibit lower average surprizal and lower token-rank variance when scored by a sufficiently capable reference model, a systematic difference that probability-based features are designed to capture and that complements the purely linguistic statistics discussed in earlier sections.

4.8. Perplexity and Probability-Based Metrics

Perplexity is the canonical metric for evaluating how well a language model predicts a held-out sequence. As shown in Equation 34, it is defined as the exponential of the average negative log-likelihood per token, so that a model assigning uniformly high probability to each observed token yields low perplexity, while a model that is frequently surprised by the observed tokens yields high perplexity.

$$\mathrm{PPL}(w_1, \ldots, w_N) = \exp\left(-\frac{1}{N} \sum_{i=1}^{N} \log P(w_i \mid w_1, \ldots, w_{i-1}) \right) \tag{34}$$

Intuitively, the value of Equation 34 may be read as the effective branching factor of the sequence under the model, meaning a perplexity of 10 means that the model is, on average, as uncertain at each step as if it were choosing uniformly among 10 equally likely continuations (Jurafsky and Martin, 2026).

Because LLMs generate text by sampling from their own high-probability regions, passages they produce tend to achieve lower perplexity under a reference model than

human text does, which forms the foundation of straightforward threshold-based detectors (Schaaff et al., 2023). This approach is effective when the reference model is closely related to the generator, but it degrades when the two diverge. Meaning, text generated by a model that the reference scorer has not encountered will not necessarily appear low-perplexity, and highly technical or formulaic human writing may score deceptively low. These limitations mean that raw perplexity alone is an unreliable detector and must be combined with other signals to achieve robust performance across diverse text types and generation methods.

5. Large Language Models

LLMs represent an important development in NLP, as they are capable of producing fluent, coherent text that increasingly resembles human writing across a broad range of domains and styles. The architectural backbone of these models, namely the encoder, decoder, and encoder-decoder Transformer variants, has already been introduced in Chapter 3, and the distributional and stylometric signatures that distinguish their output from human writing are treated in Chapter 4. This chapter focuses on the training paradigms and scaled instantiations that turn the bare architecture into an LLM. It begins by examining pre-training paradigms and the objectives that shape model behaviour, following the masked and autoregressive training strategies introduced by Devlin et al. (2019) and scaled by Brown et al. (2020). It then addresses transfer learning and fine-tuning as the mechanisms through which general-purpose models are adapted to specific tasks. It closes by surveying the representative LLM families relevant to this master's thesis, distinguishing those used as detector backbones from those targeted as generators.

5.1. Pre-Training Paradigms and Objectives

Pre-training refers to the process of learning general-purpose language representations from large unlabelled corpora before any task-specific supervision is applied. Prior to this paradigm, NLP systems were largely built around task-specific architectures trained from scratch on relatively small labelled datasets (Jurafsky and Martin, 2026), which limited both their generalization ability and their data efficiency. The shift toward pre-trained models fundamentally changed this landscape by allowing a single model to acquire broad linguistic knowledge during an initial training phase, which could then be transferred to downstream tasks with rather little additional changes needed. Devlin et al. (2019) demonstrated that this approach yielded substantial gains across a wide range of benchmarks with the BERT model, and Brown et al. (2020) later showed that scaling pre-trained models to hundreds of billions of parameters allowed them to perform many tasks with no task-specific training at all, relying solely on natural language prompts with their GPT models.

One example of a common pre-training objective is masked language modelling, introduced by Devlin et al. (2019) and already described in Chapter 2, where

its loss is given in Equation 1. In this formulation, a proportion of the input tokens are randomly replaced with a special mask token, and the model is trained to reconstruct the original tokens from the surrounding bidirectional context. Because the model must draw on both left and right context simultaneously, it is encouraged to build rich, contextually sensitive representations for every position in the sequence. This bidirectional training signal is particularly well suited to tasks that require understanding rather than generation, since the model learns to integrate contextual information from the entire input rather than predicting tokens in a fixed left-to-right order.

In contrast, generative models such as the GPT family adopt a causal or autoregressive language modelling objective, in which the model is trained to predict each token given only the tokens that precede it (Brown et al., 2020; Radford et al., 2019a). This left-to-right approach is a natural fit for text generation, since producing a sequence token by token mirrors the training objective directly. Given a sequence $\mathbf{x} = (x_1, \ldots, x_T)$, the causal language modelling objective maximizes the quantity given in Equation 35, which sums the log-likelihood of each token conditioned on all preceding tokens.

$$\mathcal{L}_{\text{CLM}} = \sum_{t=1}^{T} \log P(x_t \mid x_1, \ldots, x_{t-1}; \theta) \tag{35}$$

Although the unidirectional context means that causal models cannot condition on future tokens, this constraint is immaterial during generation, where future tokens have not yet been produced. The result is a class of models that can generate extended, coherent text sequences.

5.2. Transfer Learning and Fine-Tuning

Transfer learning is the practice of reusing representations acquired during pre-training as a starting point for downstream tasks, rather than initializing a model from scratch for each new problem. The intuition behind this approach is that pre-training on large unlabelled corpora causes a model to internalize general syntactic, semantic, and world-knowledge representations that remain broadly useful regardless of the specific task to which the model is later applied (Pan and Yang, 2009). Because these representations already encode substantial linguistic structure, adapting them to a supervised task typically requires far less labelled data than training from random initialization would demand. Consequently, transfer learning is commonly implemented in practice through a process called fine-tuning, where a pre-trained model is fine-tuned on a given downstream task without the need of full reinitialization and pre-training. Devlin et al. (2019) demonstrated

this concretely by showing that fine-tuning a single pre-trained model consistently achieved state-of-the-art results across eleven diverse natural language processing benchmarks, each with only modest amounts of task-specific supervision.

A common fine-tuning approach involves appending a task-specific head to the pre-trained model and then updating all parameters of that head on the labelled training data. For binary classification tasks such as distinguishing human-written from LLM-generated text, the training objective is empirical risk minimization with a binary cross-entropy loss, as formalized in Equation 2. Howard and Ruder (2018) observed that fine-tuning all layers simultaneously with a uniform learning rate tends to degrade lower-level representations that were carefully shaped during pre-training, and instead proposed discriminative learning rates that decrease with network depth. This sensitivity to learning rate reflects a broader concern about catastrophic forgetting, an issue where aggressive parameter updates on task-specific data overwrite general representations that took large-scale pre-training to acquire (Devlin et al., 2019). Careful scheduling and regularization are therefore standard practice when adapting large pre-trained models to downstream tasks.

Given the computational cost of updating all parameters in large models, Parameter-Efficient Fine-Tuning (PEFT) methods have emerged as a practical alternative (Ding et al., 2023). Rather than modifying the full set of pre-trained weights, these approaches keep the model's backbone frozen and introduce small trainable modules that absorb task-specific adaptation. Houlsby et al. (2019) proposed inserting compact adapter modules between Transformer layers, adding only a small fraction of additional parameters while achieving performance competitive with full fine-tuning. Hu et al. (2022) introduced Low-Rank Adaptation (LoRA), in which weight updates are constrained to a low-rank decomposition, expressing the update to a weight matrix $W \in \mathbb{R}^{d \times k}$ as the product of two smaller matrices $A \in \mathbb{R}^{d \times r}$ and $B \in \mathbb{R}^{r \times k}$ with rank $r \ll \min(d, k)$, so that only A and B are trained while the original weights remain fixed. Beyond reducing the number of trainable parameters, these methods preserve the general representations encoded during pre-training more reliably than full fine-tuning, which is particularly valuable when labelled data is scarce.

5.3. Representative LLM Architectures and Models

The three Transformer variants described in Chapter 3, namely encoder-only, decoder-only, and encoder-decoder, are also the structural templates from which the LLMs relevant to this master's thesis are built. The aim of this section is therefore not to re-derive the architectural properties of each variant but to identify

the specific examples that recur in the detection literature and to clarify the role each plays in present work.

Within the encoder-only family, BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019b), and DeBERTa (He et al., 2020) are the canonical reference models. BERT was the model that popularized the combination of a Transformer encoder with the masked language modelling objective introduced in Chapter 2, demonstrating that a single pre-trained encoder could be fine-tuned to reach state-of-the-art performance on a wide range of natural language understanding benchmarks. RoBERTa revisited several of the design choices in the original BERT recipe and reported empirical gains from training on more data and for more epochs, using larger mini-batches, removing the next-sentence prediction auxiliary objective, and dynamically changing the masking pattern across epochs. DeBERTa (He et al., 2020) extended the encoder family further with disentangled attention over content and position together with an enhanced mask decoder, yielding further improvements on understanding benchmarks. Of these three models, RoBERTa has become the common model of choice for supervised LLM-generated text detectors because its bidirectional contextual representations expose a fixed-length classifier head to information drawn from the entire input simultaneously. This property is the same one that motivates its use as the Transformer encoder in the hybrid architecture developed in this master's thesis.

Within the decoder-only family, the GPT series (Radford et al., 2019a; Brown et al., 2020) made autoregressive generation a standard approach for open-ended text production, and the open-weight LLaMA family (Touvron et al., 2023) later made similar capabilities available to a wider research audience at smaller parameter counts. These models are the main generators whose outputs this master's thesis aims to detect, and their fluency is one reason their outputs can be hard to distinguish from human writing.

Encoder-decoder models such as T5 (Raffel et al., 2020) and BART (Lewis et al., 2020) are relevant to the detection problem in two distinct ways. Their outputs are themselves candidates for detection when these models are used to generate or summarize text, and they appear in adversarial pipelines in which a paraphrase model is used to rewrite LLM-generated text in an attempt to evade detection, a possible threat that the hybrid architecture developed in this master's thesis is designed to address.

These three families occupy complementing roles in the present work. The master's thesis targets text produced by decoder-only and encoder-decoder models, since these are the generative architectures most widely deployed in practice and most frequently studied in the detection literature (He et al., 2024b), while the detector itself is built on encoder representations. Tufts et al. (2025) showed that detectors trained against one generative model family do not always transfer

reliably to others, which motivates augmenting encoder representations with model-agnostic lexical, syntactic, and stylometric features that do not depend on the specific generating architecture. This framing, encoders for detection and decoders as the detection target, motivates the architectural choices described in Chapter 8.

6. Detection of LLM-Generated Text

This chapter examines how LLM-generated text can be distinguished from human writing in practice. Chapter 5 described the training paradigms and model families that produce the text in question, and Chapter 4 described the linguistic, stylometric, and probability-based signals that this text tends to carry. The aim now is to show how those signals can be turned into a working detector. The chapter first formulates detection as a binary classification problem and compares the two main approaches, zero-shot detection and supervised detection. It then discusses feature-based methods that work directly with explicit linguistic and stylometric measurements, together with watermarking and provenance schemes that embed a verifiable signal at generation time rather than recovering one afterwards. The chapter then turns to the adversarial side of the problem, presenting a taxonomy of attack strategies organized by linguistic granularity, the perturbation techniques used at each level, and the defensive measures that have been proposed against them. The final part of the chapter argues for hybrid detection architectures that combine several complementary signal sources within a single model, and introduces the gated fusion idea that underlies the architecture developed later in this master's thesis.

6.1. Problem Formulation and Detection Paradigms

The detection of LLM-generated text can be formally characterized as a binary classification problem (Maloof, 2006). Let $\mathcal{X}$ denote the space of all possible text sequences, and let P_{human} and P_{LLM} represent the probability distributions over $\mathcal{X}$ induced by human authors and language models, respectively. Given a text sample $x \in \mathcal{X}$, the detection task seeks to determine whether x was drawn from P_{human} or P_{LLM}.

In the context of supervised learning, this translates to learning a classifier model $f : \mathcal{X} \to \{0, 1\}$ from a labelled dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, where $y_i = 0$ refers to human authorship and $y_i = 1$ indicates LLM generation. The classifier is trained

to minimize the empirical risk objective defined in Equation 2 (Shalev-Shwartz and Ben-David, 2014), where $\mathcal{L}$ is typically binary cross-entropy. The learned function f is then evaluated on its ability to generalize to unseen samples from the joint distribution $P(x, y)$.

What distinguishes LLM-generated text detection from standard binary classification in ML is the adversarial nature of the problem definition. Unlike static classification tasks where class distributions remain relatively stable, the LLM-generated text distribution P_{LLM} is non-stationary and evolves with each new model architecture and generation strategy over time (He et al., 2024b; Dugan et al., 2024). Furthermore, the decision boundary between human and LLM-generated text is not determined by fixed intrinsic semantic properties, but rather by learned statistical regularities that users can actively attempt to obscure, i.e., by applying adversarial techniques such as paraphrasing, synonym substitution, or style polishing (Dugan et al., 2024; He et al., 2025).

This adversarial setting introduces two critical theoretical considerations (Tufts et al., 2025; Crothers et al., 2023). First, the detection problem exhibits a fundamental imbalance between in-distribution accuracy and out-of-distribution robustness. A detector trained on outputs from one language model family may fail catastrophically when evaluated on text from different models, as the learned decision boundary f may encode model-specific stylistic artefacts rather than generalizable cues (Tufts et al., 2025; Crothers et al., 2023). Second, users may use adversarial transformations that alter distributional signatures, creating an asymmetry between the detection and generation tasks. A detector must correctly classify all LLM-generated samples, whereas a user may only need to find a single successful evasion strategy to compromise detection reliability and accuracy (Sadasivan et al., 2023).

6.2. Zero-Shot and Supervised Detection Methods

The fundamental challenge in detecting LLM-generated text involves correctly classifying a text sample x as drawn from either the human distribution P_{human} or the LLM distribution P_{LLM}. This binary classification task can be addressed through two theoretically distinct paradigms that differ in their assumptions and computational requirements (Crothers et al., 2023).

The first paradigm, zero-shot detection, presumes that LLM-generated text shows distinctive statistical cues extracted from the generative process itself, without requiring labelled training data. This approach is grounded in the formal characterization of natural language as a stochastic process where each token w_t is

conditionally dependent on its context $w_{<t}$ (Jurafsky and Martin, 2026). The theoretical basis rests on the token probability distributions that LLMs produce through their final softmax layer. When generating text, LLMs assign probabilities to candidate tokens via $P(w_t \mid w_{<t})$, computed by passing the final hidden state through a softmax activation. LLM-generated text, shaped by model-specific decoding strategies and sampling biases, tends to favour a smaller set of highly probable tokens, resulting in a more concentrated probability distribution over the vocabulary compared to human writing.

A foundational line of work builds on this paradigm by looking at how likely individual tokens are under a model's output distribution (Gehrmann et al., 2019b). Let $p_\theta(w_t \mid w_{<t})$ denote the probability that a model with parameters θ assigns to token w_t given the previous context $w_{<t}$. Human-written text often shows word choices that fall in relatively low-probability regions of this distribution, reflecting the variability, creativity, and situational awareness of human language use. In contrast, LLM-generated text is shaped by decoding strategies and model biases that tend to favour a smaller set of highly probable tokens, leading to more concentrated probability distribution over the vocabulary. This difference is assumed to be measurable by detectors in the zero-shot paradigm.

A more theoretically grounded approach examines the geometric structure of the log-probability induced by a language model (Mitchell et al., 2023). The core idea is that LLM-generated passages tend to occupy regions of high log-probability under the generating model. That is, a generated text x is expected to have higher log-probability than nearby variants that preserve its meaning. This intuition is captured by the curvature score, as shown in Equation 36, where p denotes the source language model and q is a perturbation mechanism that produces semantically similar alternatives to x. Under this approach, LLM-generated text is expected to show $d(x; p, q) > 0$, since language models are trained to assign high probability to their outputs, making generated text more likely than semantically similar alternatives not produced by the model.

$$d(x; p, q) = \log p(x) - \mathbb{E}_{\tilde{x} \sim q(\cdot|x)} \left[\log p(\tilde{x})\right] \tag{36}$$

The main theoretical appeal and benefit of zero-shot detection lies in its model-agnostic design and relative interpretability, as the detection cues are taken directly from the output probability distributions of language models. At the same time, this paradigm relies on the strong assumption that probability-based signatures remain stable under meaning-preserving transformations, i.e., paraphrases that preserve semantic content but alter surface-level wording. In practice, this assumption often fails. When text is paraphrased without changing its meaning, its probability under a given model can shift substantially (Krishna et al., 2023). As a result, zero-shot detectors are very vulnerable to adversarial rewriting, where adversarial attacks

produce text that is semantically equivalent to LLM output but no longer exhibits the statistical patterns these detectors heavily rely on.

The second major paradigm, supervised detection, approaches the problem from a different angle. Rather than computing explicit probability-based measures, it frames human versus LLM-generated text detection as a standard ML classification task over learned representations (Crothers et al., 2023). These methods rely on neural networks to capture implicit stylistic, syntactic, and semantic cues into their hidden representations that correlate with text origin. From a theoretical standpoint, this paradigm relies on the expressive power of neural models and the inductive biases of Transformer architectures (Vaswani et al., 2017).

A supervised detector can be formalized as a function $f_{\boldsymbol{\theta}} : \mathcal{X} \to [0, 1]$, parameterized by $\boldsymbol{\theta}$, where $\mathcal{X}$ denotes the space of text sequences and the output represents the estimated probability that a text was generated by an LLM. The parameters are learned by minimizing the empirical risk objective from Equation 2 (Shalev-Shwartz and Ben-David, 2014) over a labelled dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, where $\mathcal{L}$ is typically binary cross-entropy and $y_i \in \{0, 1\}$ indicates whether a sample is human-written or LLM-generated.

The key strength of supervised detection lies in its flexibility. Because the model is not tied to predefined statistical features, it can learn complex, non-linear decision boundaries that are difficult to express analytically with just the syntactic cues. Transformer-based encoders such as RoBERTa (Liu et al., 2019b) and DeBERTa (He et al., 2020) can capture long-range dependencies and subtle stylistic signals through self-attention and neural nature of their architecture. The resulting representations go beyond surface-level token statistics and may encode higher-level properties such as discourse structure, coherence, and compositional patterns of language.

However, this paradigm also comes with clear theoretical limitations. The ability of $f_{\boldsymbol{\theta}}$ to generalize depends heavily on the coverage and diversity of the training data $\mathcal{D}$. When evaluated on text from unseen domains or from language models not represented during training, supervised detectors may rely on brittle correlations (Tufts et al., 2025). This issue can be viewed as a domain shift problem. Instead of approximating the desired distribution $P(y \mid x)$, the model may implicitly learn $P(y \mid x, \text{domain})$, reducing robustness under distributional change, including changes under adversarial attack conditions.

The difference between zero-shot and supervised detection methods highlights a broader trade-off between generalization and discriminative power. Zero-shot methods offer statistical, model-agnostic criteria but are sensitive to meaning-preserving adversarial changes of input text. Supervised approaches can achieve strong performance within known distributions, yet they may under-perform on out-of-distribution input text.

It is important to mention that recent work seeks to combine the strengths of both paradigms through adversarial learning formulations that see detection as a game between competing models (Hu et al., 2023). In this setting, a detector D and an adversarial paraphraser A are trained according to the objective depicted in Equation 37.

$$\min_{D} \max_{A} \mathbb{E}_{x \sim P_{\text{LLM}}} \left[\mathcal{L}(D(A(x)), 1) \right] + \mathbb{E}_{x \sim P_{\text{human}}} \left[\mathcal{L}(D(x), 0) \right] \tag{37}$$

This formulation pushes the detector to focus on features that remain stable under adversarial rewriting, thereby improving robustness to semantic evasion. In a similar context, ensemble-based methods that route inputs to specialized detectors across domains provide a theoretical mechanism for balancing generalization and specialization (Dugan et al., 2025). Taken together, these developments show a problem in LLM-generated text detection, lying between interpretability, robustness, and accuracy, which continues to motivate hybrid approaches that integrate statistical features with learned representations.

6.3. Feature-Based Detection Approaches

Before large pretrained models became the default tool for text classification, detection systems relied on carefully engineered feature sets extracted from candidate texts and passed to classical supervised classifiers such as support vector machines or logistic regression. These features typically combined perplexity scores from smaller language models with stylometric statistics, readability indices, and n-gram frequencies, constituting a broad if shallow characterization of the text (Schaaff et al., 2023). Such systems demonstrated that machine-generated text is distinguishable from human text even without access to deep representations, which validated the hypothesis that generation leaves measurable statistical traces at the surface level.

Feature-based detection, as an ML approach, may be looked at from the angle of an older task, namely the authorship attribution. Both tasks seek to find a feature representation that separates one source of text from another (Schaaff et al., 2023). The two tasks are not equivalent, and distinguishing human writing from increasingly fluent machine output may be considerably harder than attributing a text to one of a small set of known authors. This is precisely why robust, universally accurate detection of LLM-generated text remains to be an open problem rather than a solved one. The same signal provided by extracted linguistic features is nonetheless informative for both tasks. Language models trained on human text inherit many surface-level statistical properties of that training data but differ in subtle regularities of function word use, punctuation rhythm, and syntactic

preference that emerge from the generation process itself. The stylometric features described in Chapter 4 are therefore directly transferable to the detection setting, and their inclusion in the hybrid pipeline developed in this master's thesis is motivated by this established tradition of source attribution.

Feature-based approaches offer several practical advantages over purely neural systems. The extracted features are interpretable, the classifiers require relatively little labelled data to train, and the resulting systems tend to generalize more stably when the distribution of input text shifts across domains or model families (Crothers et al., 2023; Schaaff et al., 2023). Their principal limitation is that a fixed, manually designed feature set will inevitably fail to capture all the variation that distinguishes human from machine text, particularly as language models improve and close the surface-level gap (Schaaff et al., 2023). This motivates the hybrid architecture adopted in this master's thesis, which pairs the breadth and interpretability of linguistic features with the depth of contextual neural representations, aiming to inherit the advantages of both paradigms while mitigating their respective weaknesses.

6.4. Watermarking and Provenance

The detection approaches discussed so far analyse text after it has been generated. An alternative foundational approach takes a different angle by embedding verifiable signals directly into the generation process. This section reviews two closely related frameworks of algorithmic watermarking and cryptographic provenance.

Watermarking methods alter the generation procedure itself to introduce statistical patterns that can later be detected. Unlike post-hoc detectors, which attempt to decipher authorship from naturally occurring properties of text (Gehrmann et al., 2019b; Mitchell et al., 2023), watermarking creates an intentional and verifiable signal in the text. The underlying idea is to slightly bias the token sampling process in a way that is not noticeable to human readers but can be reliably recovered using algorithms.

A standard watermarking scheme divides the vocabulary $\mathcal{V}$ at each generation step t into two disjoint sets, a "green list" $\mathcal{G}_t \subset \mathcal{V}$ and a "red list" $\mathcal{R}_t = \mathcal{V} \setminus \mathcal{G}_t$ (Kirchenbauer et al., 2023). This partition is computed using a unique cryptographic hash of the preceding context $w_{<t}$, making the green list assignment deterministic yet unpredictable without access to the secret seed. During generation, the model's logits are adjusted by a bias $\delta > 0$ to favour tokens from the green list, as depicted in Equation 38, where $\ell_t(w)$ denotes the original logit for the token w at position t.

$$\ell'_t(w) = \begin{cases} \ell_t(w) + \delta & \text{if } w \in \mathcal{G}_t \\ \ell_t(w) & \text{if } w \in \mathcal{R}_t \end{cases} \tag{38}$$

Detecting a watermark is framed as a hypothesis testing problem. Under the null hypothesis H_0, where the text is assumed to be generated without access to the green list, the number of green-list tokens n_g in a sequence of length T follows a binomial distribution with success probability of $\gamma = |\mathcal{G}_t|/|\mathcal{V}|$. This leads to the test statistic shown in Equation 39.

$$z = \frac{n_g - \gamma T}{\sqrt{\gamma(1-\gamma)T}} \tag{39}$$

Large values of z indicate a statistically significant excess of green tokens, providing evidence that a watermark is present in the test sample, with the associated p-value quantifying the likelihood of such a deviation occurring by chance (Kirchenbauer et al., 2023). In theory, this framework offers reliable detectability while preserving text quality, as the magnitude of the bias δ can be kept small enough to avoid perceptible changes to the generated text.

A related watermarking paradigm represents watermarking schemes that adapt dynamically to the underlying token distribution (He et al., 2024a). By adjusting the bias based on distributional entropy, these methods try to remain effective even when text is paraphrased or modified through synonym substitution. While this improves robustness, it also complicates the balance between detectability and generation quality.

A conceptually distinct approach to attribution relies on cryptographic provenance rather than signals embedded in the text itself. Provenance systems, such as those standardized by the Coalition for Content Provenance and Authenticity (C2PA), shift the problem from statistical detection to cryptographic verification (Rosenthol, 2022; C2PA, 2023). The idea is to attach a digitally signed manifest to content, recording information about its origin, edits, and distribution path, thereby forming a verifiable chain of trust.

The reliability of provenance systems depends on how tightly the manifest is bound to the content. Two main binding strategies are commonly discussed. Hard binding uses cryptographic hashes to tie the manifest directly to the exact bit representation of the content (C2PA, 2023; Zhou et al., 2024). Any modification, however small, makes the hash invalid, offering strong evidence of tampering, but requiring exact preservation of the content. Soft binding, by contrast, relies on perceptual hashes or robust watermarks that remain stable under benign transformations such as re-encoding or light editing (C2PA, 2023). This approach is more flexible in practice, but it necessarily weakens some security guarantees.

Provenance frameworks typically rely on asymmetric cryptography, where content creators sign manifests with private keys and verifiers authenticate them using corresponding public keys (Zhou et al., 2024). While this shifts attribution from probabilistic inference to identity verification, it also introduces dependencies on

key management, certificate authorities, and broader trust infrastructure factors that are largely absent from purely algorithmic detection methods.

6.5. Evaluation Metrics and Datasets

Evaluating detection systems, whether classifiers, watermark detectors, or provenance validators, requires metrics that reflect the asymmetric consequences of errors. In many real-world settings, such as education or legal decision-making, false positives can be far more costly than false negatives (Elkan, 2001). From a perspective of making a classification decision, an input x should be classified as LLM-generated only when the condition in Equation 40 is met.

$$P(y = 1 \mid x) > \frac{C_{\text{FP}}}{C_{\text{FP}} + C_{\text{FN}}} \tag{40}$$

Here, C_{FP} and C_{FN} represent the costs of false positives and false negatives, respectively (Elkan, 2001).

As a result, simple accuracy is often misleading for detection tasks, particularly under class imbalance. Theoretical and empirical work therefore emphasizes alternative metrics that better capture performance trade-offs. Balanced accuracy assigns equal weight to both classes by averaging the true positive and true negative rates (Brodersen et al., 2010). Macro-averaged F1 computes precision and recall separately for each class before averaging, ensuring that human and LLM-generated text contribute equally (Sokolova and Lapalme, 2009; Farhadpour et al., 2024). Informedness, or ΔP, measures how much a detector's predictions improve over random guessing, offering a chance-corrected view of discriminative ability (Powers, 2020).

Benchmark datasets such as RAID (Dugan et al., 2024), MGTBench (He et al., 2024b), and M4GT-Bench (Wang et al., 2024a) put these evaluation principles into practice. RAID covers over six million generations spanning eleven LLM families, eight domains, and eleven adversarial attack types, making it one of the most comprehensive benchmarks for robustness evaluation. MGTBench provides a multi-domain benchmark with fine-grained attribution labels and focuses on cross-model detection scenarios. M4GT-Bench is a multilingual, multi-domain benchmark covering binary detection, model attribution, and mixed human-machine text identification. These benchmarks consistently highlight gaps between in-distribution performance and robustness under distribution shift, exposing weaknesses such as sensitivity to character-level noise, homoglyph substitutions, and cross-model generalization failures. Overall, the theory of evaluation in LLM-generated text detection emphasizes not only aggregate metrics, but also systematic robustness analysis under realistic and adversarial transformations.

6.6. Adversarial Pressure on Detection

The ability to distinguish machine-generated text from human writing is not a static classification problem but an active adversarial battleground. As detection systems grow more capable, those with an interest in evading them refine their strategies, producing an ongoing cycle in which advances on one side provoke countermeasures on the other. This dynamic mirrors the broader adversarial arms race observed across many areas of machine learning, where classifiers trained on one distribution of inputs are routinely challenged by purposefully constructed examples designed to sit just outside the boundary the classifier has learned (Goyal et al., 2023). In the context of LLM-generated text detection, the adversary is not necessarily a sophisticated attacker in the traditional security sense; it may simply be a user who applies paraphrasing tools, inserts deliberate stylistic variation, or mixes their own prose with model output, all of which are sufficient to undermine detectors that rely on narrow surface-level signals. Understanding the structure of these threats is therefore a prerequisite for building detection systems that remain useful under realistic conditions. The remainder of this chapter examines this challenge along two complementary lines. The next several subsections develop a taxonomy of the principal attack strategies that have been documented in the literature, covering both post-hoc perturbation methods and generation-time interventions that target the statistical and linguistic regularities that detectors exploit. The final sections then examine hybrid defence architectures, which combine heterogeneous feature sources and model components in order to distribute the evidentiary burden across signals that are harder to suppress simultaneously (Knudsen and Hardmeier, 2025). Taken together, these two threads motivate the architectural choices made in the detection system developed in this master's thesis, where the fusion of neural and lexical evidence is understood not merely as a performance strategy but as a principled response to the fragility of single-signal approaches under adversarial pressure.

6.6.1. Taxonomy of Adversarial Attacks on Text

Adversarial attacks in the natural language processing setting refer to inputs that have been deliberately constructed or modified so that a classifier produces an incorrect output, while the underlying intent or meaning of the text is preserved well enough to serve the attacker's communicative purpose. The concept was formalized in the image domain, where Goodfellow et al. (2015) demonstrated that small, carefully chosen perturbations applied to pixel values could reliably cause neural networks to misclassify images with high confidence, even when the perturbations were imperceptible to human observers. Transferring this idea to text introduces a fundamental complication. Since language is discrete rather than continuous,

meaning that perturbations must operate on tokens rather than real-valued coordinates, and even minor changes to a word or character can produce outputs that are grammatically ill-formed or semantically incoherent. This constraint shapes the entire threat landscape for text-based classifiers, because an attacker must find perturbations that are simultaneously effective against the detector, acceptable to a human reader, and feasible to apply without access to the model's internal parameters (Goyal et al., 2023). The result is a more constrained but in many respects more practically dangerous problem than its image counterpart, because the attacks that survive these constraints tend to exploit genuine properties of language that classifiers have failed to generalize over.

The literature on adversarial attacks against text classifiers has converged on a three-level taxonomy organized by the granularity of the linguistic unit that is modified (Goyal et al., 2023; Crothers et al., 2023). Character-level attacks operate on individual characters within tokens, introducing substitutions, deletions, transpositions, or homoglyph replacements that alter the surface form of a word without changing its perceived meaning for a fluent reader. Word-level attacks operate on whole tokens, replacing individual words with synonyms, paraphrases, or semantically related alternatives that preserve the local meaning of a sentence while shifting its statistical profile. Sentence and semantic-level attacks operate at a coarser granularity, restructuring entire phrases or passages through paraphrasing, back-translation, or prompt-based rewriting, often with the explicit goal of suppressing the distributional signatures that neural detectors rely on. This taxonomy is not merely a descriptive convenience; it reflects a meaningful difference in the detection signal each attack family targets, and it maps onto the architecture of the hybrid detection system developed in this master's thesis in a direct way. Character-level attacks primarily challenge lexical and surface-form features, word-level attacks pressure vocabulary statistics and embedding-based representations, and semantic-level attacks are designed to evade the deeper neural signals that Transformer-based components capture.

6.6.2. Character-Level and Word-Level Perturbations

Character-level attacks represent the most granular form of adversarial manipulation available to an attacker, and their effectiveness derives less from linguistic sophistication than from the way modern tokenization pipelines process surface form. Common techniques include homoglyph substitution, in which visually similar characters from different Unicode ranges replace standard Latin letters (Eger et al., 2019), keyboard-proximity typos that introduce plausible spelling errors (Gao et al., 2018), the insertion of zero-width spaces or other invisible characters between or within tokens (Boucher et al., 2022), and case alternation that disrupts the normalization assumptions built into many preprocessing pipelines (Dugan

et al., 2024; He et al., 2025). Each of these strategies exploits a structural property of subword tokenization. Because tokenizers such as byte-pair encoding segment words based on frequency statistics computed over clean training corpora, an unexpected character sequence will either produce an out-of-vocabulary token or force a decomposition into subword units that were never associated with the target word during pre-training, corrupting the embedding that the downstream classifier receives. Ebrahimi et al. (2018) formalized this intuition by introducing HotFlip, a gradient-based method that identifies which character-level flips produce the largest change in classifier output, demonstrating that targeted single-character edits were sufficient to cause misclassification in several text classification settings. The broader robustness implications of character noise were examined by Belinkov and Bisk (2018), who showed that neural sequence models trained on clean text degrade substantially when exposed to synthetic and natural character-level corruption, even when the errors are mild enough to be transparent to human readers. For a hybrid detector that includes lexical features computed over surface tokens, character-level perturbations present a particular challenge because they can cause a single semantically coherent word to fragment into multiple subword pieces, inflating type counts and distorting vocabulary statistics in ways that mimic properties of genuine human writing.

Word-level attacks operate at a higher granularity and generally demand more computational effort from the attacker, but they are capable of producing text that is both fluent and semantically coherent while substantially reducing classifier confidence. The dominant paradigm in this family involves synonym substitution guided by semantic similarity constraints. Candidate replacement words are drawn from a neighbourhood defined by embedding distance or WordNet relations, and substitutions are accepted only when they preserve grammaticality and leave the overall meaning sufficiently intact to satisfy a human evaluator. The TextFooler method proposed by Jin et al. (2020) instantiates this approach using a BERT-based similarity filter to screen candidate replacements, and demonstrated that the resulting adversarial examples transferred effectively across several classifiers while remaining largely indistinguishable from the original text in human evaluation. Earlier work by Alzantot et al. (2018) applied a genetic algorithm to explore the space of word-level substitutions, similarly showing that population-based search over synonym candidates could produce natural-looking adversarial inputs without access to classifier gradients. Beyond synonym substitution, attackers may also delete low-information function words or insert semantically neutral filler tokens to shift the frequency profile of a passage. These manipulations are directly relevant to hybrid detection systems that incorporate lexical features such as vocabulary richness, type-token ratios, or function word distributions, because even moderate synonym substitution can move a passage's vocabulary profile toward the region

occupied by human writers, selectively degrading one of the feature channels the detector relies upon while leaving others, such as syntactic regularity or perplexity under a fixed language model, comparatively intact.

6.6.3. Paraphrasing and Semantic Attacks

Paraphrasing attacks operate at the level of whole sentences or passages, substantially rewriting the surface form of a text while preserving its propositional content (Krishna et al., 2023). Unlike character- and word-level perturbations, which modify isolated units and risk introducing local incoherence, paraphrasing attacks produce outputs that are globally fluent and internally consistent, because the transformation is applied at a level of granularity that respects syntactic and discourse structure. In practice, attackers have access to two principal mechanisms for producing such neural paraphrases and back-translation pipelines. Neural paraphrasers trained on large parallel corpora, which learn to generate alternative surface realizations of a given meaning (Krishna et al., 2023), and Back-translation pipelines, which translate a passage into an intermediate language and then translate it back, exploiting the divergence between translation systems to introduce lexical and structural variation (Federmann et al., 2019). Both mechanisms are effective precisely because they disrupt the distributional fingerprints that perplexity-based and n-gram-based detectors rely upon. A language model assigns low perplexity to sequences whose token transitions are consistent with its training distribution, and text generated by a given model tends to follow characteristic transition patterns; paraphrasing breaks these patterns by substituting alternative phrasings that are equally natural but statistically less predictable under the original model. Sadasivan et al. (2023) provided theoretical grounding for this vulnerability, arguing that under a sufficiently capable paraphraser, the statistical distance between human and machine text distributions collapses to a point where reliable detection becomes fundamentally intractable without additional information.

A structured variant of the paraphrasing attack targets syntactic form rather than lexical content specifically, leaving the vocabulary of a passage largely intact while reorganizing its grammatical architecture (Iyyer et al., 2018). Transformations of this kind include converting active constructions to passive ones, reordering subordinate clauses, fronting adverbial phrases, and splitting or merging sentences at coordination boundaries (Iyyer et al., 2018). Because these operations alter the dependency structure of a passage without changing its word inventory, they are largely invisible to detectors that operate over bag-of-words representations or surface vocabulary statistics, but they directly perturb the syntactic feature distributions that more linguistically informed detectors rely upon. Iyyer et al. (2018) developed a method for generating syntactically controlled paraphrases using a structural constraint model, and showed that the resulting adversarial examples

successfully evaded classifiers that were robust against lexical substitution attacks, illustrating that syntactic form carries independent discriminative information that must be addressed separately. This observation has direct implications for the hybrid architecture developed in this master's thesis. By fusing a contextual neural representation with an explicit linguistic feature vector through a learned per-sample gate, the architecture is designed to ensure that no single feature channel can be neutralized in isolation. An attacker who suppresses syntactic regularity signals while preserving lexical statistics will still leave a residual signal in the neural stream, while an attacker who paraphrases at the semantic level may degrade the neural representation but leave surface-level lexical and stylometric statistics largely intact. The gate then learns to up-weight whichever stream remains informative for a given input, distributing the evidentiary basis for detection across signals that are considerably harder to manipulate simultaneously than either stream alone.

6.6.4. Perplexity Manipulation and Style Transfer

Perplexity-based detectors operate on the assumption that machine-generated text will exhibit characteristically low perplexity under a reference language model, reflecting the tendency of generators to favour high-probability token sequences (Gehrmann et al., 2019b; Mitchell et al., 2023). This assumption is exploitable, because an adversary with access to any language model can iteratively replace tokens in generated text with alternatives that push the passage perplexity into the range typical of human writing. Without requiring any knowledge of the detector's internal parameters. The attack proceeds by treating the reference model's probability assignments as a surrogate signal, selecting substitutions that raise local token surprizal while preserving fluency (Jin et al., 2020). As Sadasivan et al. (2023) demonstrated, this class of manipulation is sufficient to collapse the separability of human and machine text distributions for detectors that treat perplexity as a primary signal, and Goyal et al. (2023) situate such score-targeting strategies within the broader landscape of evasion attacks, noting that the availability of open-weight language models makes surrogate-guided manipulation practically accessible to a wide range of adversaries. The implication for the detection architecture developed in this master's thesis is that perplexity should be treated as one signal among several rather than as a reliable primary discriminator.

Style transfer attacks take a different approach, targeting the generative process itself rather than post-processing the output. Rather than modifying text after the fact, a style transfer system conditions or fine-tunes a language model to produce text that matches the surface and statistical characteristics of a specific authorial style or register, thereby suppressing the regularities that stylometric detectors exploit (Stamatatos, 2009). The effect is to move generated text away from the generic statistical profile of a large language model and toward the

idiosyncratic profile of a target author or writing community. Shen et al. (2017) demonstrated that disentangling content from style in neural text generation is achievable without parallel training data, establishing a practical foundation for this class of attack. The vulnerability this creates for stylometric detection is significant. As Stamatatos (2009) notes, authorship attribution methods are sensitive to the precise configuration of stylistic signals, and a generator that has been adapted to suppress or mimic these signals can undermine the feature branch of a hybrid detector that relies on function word frequencies, punctuation patterns, or sentence length distributions. Within the architecture proposed in this master's thesis, this motivates treating stylometric features as inputs to a cross-attended joint representation rather than as standalone classifiers, so that their degradation under style transfer is partially compensated by evidence from the neural and syntactic streams.

6.7. Adversarial Training and Defence Mechanisms

The dominant framework for building classifiers that resist adversarial perturbations is adversarial training. Instead of training the model only on the original examples, it also confronts the model during training with the most damaging perturbed version of each example, so that the model learns to classify correctly even in the worst case allowed by a defined perturbation budget (Madry et al., 2018). The resulting optimization problem, referenced here as Equation 41, seeks model parameters θ that minimize the expected loss under the most damaging perturbation δ within a budget $\mathcal{S}$:

$$\min_{\theta} \; \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\max_{\delta\in\mathcal{S}} \mathcal{L}(f_\theta(x+\delta), y) \right] \tag{41}$$

Madry et al. (2018) instantiated this framework using projected gradient descent to approximate the inner maximization, producing models with substantially improved robustness to norm-bounded input perturbations in the image domain. Miyato et al. (2017) adapted the same principle to text classification by computing adversarial perturbations in the continuous embedding space rather than over discrete tokens, demonstrating improved generalization alongside robustness gains. The core insight shared by both approaches is that exposing the model to hard examples during training forces it to learn decision boundaries that do not rely on fragile, locally inconsistent features.

Several complementary defence strategies have been proposed alongside adversarial training. Input preprocessing methods attempt to reverse or neutralize

character-level noise before it reaches the classifier, for instance by applying spell-correction or Unicode normalization as a first-pass filter (Pruthi et al., 2019). Two-stage detection pipelines interpose a separate adversarial example detector before the main classifier, flagging suspicious inputs rather than attempting to classify them directly (Zhou et al., 2019). Certified robustness methods offer formal guarantees that a classifier's output will not change for any perturbation drawn from a bounded set of word substitutions, though these guarantees currently scale poorly to large models and realistic perturbation sets (Jia et al., 2019). As Goyal et al. (2023) notes, adversarial training itself carries a significant cost. Since it increases training time substantially and tends to reduce accuracy on clean, unperturbed examples, a trade-off that becomes increasingly difficult to manage as model and dataset size grow. The detection system developed in this master's thesis does not rely on adversarial retraining to achieve robustness; instead, it addresses the problem through feature complementarity, distributing the classification signal across lexical, syntactic, stylometric, and neural channels so that perturbations capable of suppressing any single channel are unlikely to suppress all simultaneously.

6.8. Robustness Evaluation Methodologies

Evaluating adversarial robustness requires a more carefully specified protocol than standard classification evaluation, because the results depend heavily on the assumed threat model. The two principal threat models are white-box evaluation, in which the attacker has full access to the classifier's parameters and can compute gradients directly, and black-box evaluation, in which the attacker can only query the classifier and must rely on transferability or surrogate models. Attack success rate and accuracy under attack are the standard reported metrics, but neither is meaningful without a precise statement of which threat model was assumed, what perturbation budget was permitted, and which attack algorithm was used to generate adversarial examples. Morris et al. (2020) addressed the resulting reproducibility problem by introducing TextAttack, a unified framework that standardizes the implementation of a wide range of text adversarial attacks and allows results to be compared across models and datasets under controlled conditions. Goyal et al. (2023) survey the benchmarking landscape more broadly and find that inconsistent threat model specification remains one of the primary sources of incomparability across papers in this area.

Current evaluation practice carries several well-recognised limitations. Most published robustness results are reported against a small set of canonical attack methods, meaning that a model that appears robust under those conditions may not generalize to novel attack strategies (Goyal et al., 2023). Standardized holdout corpora for adversarial robustness evaluation are largely absent from the field,

making it difficult to assess whether improvements reflect genuine robustness gains or overfitting to a specific attack family. Semantic equivalence verification, which is necessary to confirm that adversarial examples actually preserve the original meaning, typically requires human annotation and represents a significant bottleneck for large-scale evaluation (Morris et al., 2020). The evaluation conducted in this master's thesis adopts the black-box threat model as the most realistic scenario for deployed detection systems, where an attacker has no access to model internals and can only observe classifier outputs. Both clean accuracy and accuracy under attack will be reported throughout, so that any robustness gains achieved by the hybrid architecture are assessed against the clean performance cost they impose.

6.9. Combining Neural and Feature-Based Approaches

Purely neural detectors, despite their strong performance on in-distribution benchmarks, exhibit a well-documented fragility under distribution shift and adversarial perturbation (Sadasivan et al., 2023; Krishna et al., 2023). Because they learn to compress all discriminative information into a single dense representation, a perturbation that moves the input across the boundary of that representation is sufficient to cause misclassification, and there is no independent signal available to contradict the corrupted neural output. Feature-based detectors avoid this single-point-of-failure property by grounding classification in interpretable, independently computable signals, but they lack the representational capacity to capture the subtle generative artefacts that emerge from large language models and that are not straightforwardly expressible in terms of lexical counts or syntactic distributions. This issue is why a hybrid approach may be beneficial for solving this problem. Neural and feature-based signals are sensitive to different properties of the input, so their combination provides coverage that neither category achieves alone (Knudsen and Hardmeier, 2025). The lexical, syntactic, stylometric, and language-model-based features described in Chapter 4 each capture a distinct dimension of the distributional difference between human and machine text, and a hybrid architecture is designed to ensure that suppressing any one of these dimensions does not eliminate the classification signal entirely.

The general structural template shared by neural-feature hybrid detectors in the detection literature positions a pre-trained Transformer encoder and a handcrafted feature branch as parallel processing streams, whose outputs are combined before a final classification head (Yadagiri et al., 2024; Knudsen and Hardmeier, 2025). The neural encoder produces contextualized token representations that capture

long-range syntactic and semantic dependencies, while the feature branch computes fixed-dimensional vectors encoding lexical richness, syntactic dependency profiles, stylometric markers, and perplexity estimates. Concatenation and learned linear projection are the most common fusion strategies in the literature, but they treat the two streams as independent and do not allow each to inform the other's internal representations. The architecture developed in this master's thesis departs from this template at precisely that point. Rather than combining the two streams through simple concatenation, it employs a gated fusion mechanism in which a learned per-sample gate adaptively weighs the contribution of the neural and the linguistic representations before the classification decision is made. The motivation for this design and its formal specification are presented in Chapter 8.

6.9.1. Feature Fusion Strategies

Feature fusion in the context of hybrid detection refers to the problem of combining heterogeneous representations, specifically the dense contextualized vectors produced by a neural encoder and the sparse or scalar handcrafted feature vectors computed by a linguistic feature branch, into a unified input for a downstream classifier (Baltrušaitis et al., 2018). The design space for this combination has three principal dimensions. These can be named as where in the processing pipeline fusion occurs, how the representations are aligned in dimensionality so that they can be processed jointly, and whether the weights governing the combination are fixed by hand or learned from data. Baltrušaitis et al. (2018) identify these dimensions as the central axes along which multimodal fusion architectures differ, and the same framework applies directly to the text detection setting, where the two modalities are the neural and feature-based representations of the same input passage. Knudsen and Hardmeier (2025) demonstrate that the specific choice of fusion strategy has a measurable effect on detection robustness, motivating careful consideration of the available paradigms rather than defaulting to the simplest combination method.

Early fusion concatenates the outputs of all feature branches at the input level, before any shared processing takes place (Baltrušaitis et al., 2018). This approach allows a single model to learn interactions between feature types from the start of the computation, but it requires the model to reconcile representations that differ substantially in scale, density, and dimensionality from the outset, which typically demands careful normalization and can lead to the neural component dominating the combined representation simply by virtue of its higher dimensionality. Late fusion takes the opposite approach, allowing each branch to produce its own classification output independently, and then combining those outputs through voting, averaging, or a learned meta-classifier (Wolpert, 1992). This paradigm is modular and easy to extend, since new branches can be added without modifying existing ones,

but branch representations cannot inform one another at any intermediate stage, which forecloses the possibility of synergistic learning where evidence in one stream sharpens the representations in another (Goyal et al., 2023). Intermediate fusion resolves this tension by merging the branch representations at an internal layer, after some branch-specific processing has occurred but before the final classification decision. Because the branches retain some capacity for independent computation before fusion, intermediate fusion is generally better suited to settings where the feature streams carry complementary rather than redundant information, since each stream can develop a representation that reflects its own structure before being integrated with the other. The gated fusion mechanism described next is an instance of learned intermediate fusion, in which a per-sample gate controls the relative contribution of each stream before the classification head is applied.

Gated Fusion and Adaptive Integration

A fixed-weight combination of branch representations assumes that the relative reliability of each feature stream is constant across all inputs, an assumption that does not hold in practice (Knudsen and Hardmeier, 2025). For a given passage, the neural representation may carry most of the discriminative signal, while for another, stylometric or syntactic features may be more informative because the neural signal has been partially suppressed by paraphrasing or style transfer (Krishna et al., 2023). Gating mechanisms address this by learning to weight the contribution of each branch on a per-sample basis (Chaudhari et al., 2021). The general form of such a learned gate, in which two feature vectors are combined through an element-wise convex combination whose mixing weights are computed from the concatenation of the two inputs, has already been introduced in Equation 33 of Chapter 3. In the hybrid detection setting, the same construction is applied with $\mathbf{f}_A$ playing the role of the pooled Transformer representation and $\mathbf{f}_B$ playing the role of the projected linguistic feature vector, so that the gate vector decides for each feature dimension whether the dense neural signal or the explicit linguistic signal is more reliable for the sample at hand (Knudsen and Hardmeier, 2025). This formulation is directly analogous to the input and forget gates of the LSTM cell introduced by Hochreiter and Schmidhuber (1997), in which learned per-dimension gates operating over hidden state vectors provide an effective mechanism for selectively retaining or suppressing information.

Beyond the binary gating formulation, several adaptive integration schemes have been proposed that extend this idea to settings with more than two branches or where the optimal integration policy varies more continuously with input content. Mixture-of-experts weighting assigns a learned softmax distribution over branches, with each branch conceptualized as a specialist whose contribution is scaled by a gating network's confidence in its relevance (Shazeer et al., 2017). Attention-based

pooling computes a weighted average over branch output vectors using attention scores derived from a shared query representation, allowing the model to attend differentially to different parts of each branch's output rather than summarizing it with a single gate value (Lin et al., 2017). Across these variants, the consistent finding in the hybrid detection literature is that learned adaptive weighting outperforms fixed combination weights under distribution shift, precisely because the optimal balance between feature streams changes as the input distribution deviates from the training regime (Knudsen and Hardmeier, 2025). Since distribution shift driven by adversarial perturbation is the primary robustness challenge here, the gated fusion mechanism is designed to operate adaptively per sample, ensuring that no fixed weighting assumption can be exploited by an attacker who targets a specific feature stream.

6.9.2. Ensemble Methods and Model Combination

Ensemble methods and within-model hybrid fusion both pursue complementarity among heterogeneous signals, but they differ in where that complementarity is realized. Hybrid fusion integrates representations from different feature streams inside a single model during a shared training process, while ensemble methods combine the outputs of multiple independently trained models at inference time, with each base model developing its own internal representations without direct interaction with the others. The principal ensemble strategies are bagging, which trains multiple base models on resampled subsets of the training data and averages their predictions to reduce variance (Breiman, 1996), boosting, which trains models sequentially so that each focuses on the examples that the previous model misclassified (Freund and Schapire, 1997), and stacking, which trains a meta-classifier on the output distributions of several base models rather than on the raw input features. In NLP classification settings, the robustness benefit of ensembling derives primarily from diversity among the base models. If individual classifiers are sensitive to different features and make uncorrelated errors, their combination suppresses individual failure modes and produces a more stable aggregate decision (Goyal et al., 2023).

Stacking is the most directly relevant ensemble variant for LLM-generated text detection, because it allows a meta-classifier to learn which base detectors are more reliable under which input conditions, rather than simply averaging their votes (Breiman, 1996). In a typical stacked detection system, several base detectors, perhaps one perplexity-based, one neural, and one stylometric, each produce a probability score, and a lightweight meta-classifier is trained on those scores to produce the final prediction (Wolpert, 1992). While this approach can capture complementarity across detector types, it introduces substantial practical costs. Since each base model must be trained and stored independently, inference requires

a full forward pass through every base model, and the meta-classifier requires its own held-out training set to avoid overfitting to the base models' idiosyncrasies. The hybrid architecture developed in this master's thesis achieves ensemble-like complementarity among lexical, syntactic, stylometric, and neural signals within a single end-to-end trainable model, with a shared gradient signal ensuring that all feature branches are jointly optimized for the detection objective. This design retains the coverage advantages of multi-source evidence while incurring only the inference cost of a single model, making it substantially more practical to deploy than a stacked ensemble of comparable expressive power.

6.9.3. Pooling Strategies for Sequence Representations

Transformer encoders produce a matrix of contextualized token representations, one vector per input token, but a document-level classifier requires a single fixed-length vector as input. Pooling may be named the operation that bridges these two levels of representation, and the choice of pooling strategy has a measurable effect on classification performance. The approaches that may be used are CLS token extraction, mean pooling, and max pooling. The CLS token, introduced by Devlin et al. (2019) as a special prepended symbol whose final hidden state is intended to aggregate sequence-level information, is the natural choice when the pre-training objective has explicitly trained this token to serve as a summary representation, as is the case in BERT's next-sentence prediction task. However, when a model is used in a domain or task that differs from its pre-training setup, the CLS token's aggregation capacity is not guaranteed, and mean pooling over all token states frequently produces stronger document representations in such settings. Reimers and Gurevych (2019) demonstrated this empirically in the context of sentence embedding, showing that mean-pooled representations from BERT consistently outperformed CLS-based representations on semantic similarity benchmarks, and attributed this to the fact that the CLS token's pre-training signal does not encourage it to encode fine-grained token-level content uniformly.

Attention-based pooling provides a more flexible alternative by replacing the uniform average with a learned weighted sum, where the weight assigned to each token is computed from its own representation (Lin et al., 2017). Concretely, a trainable query vector $\mathbf{q}$ is compared against each token hidden state $\mathbf{h}_t$ to produce a scalar attention score, which is then normalized across the sequence and used to compute the pooled output. Referencing this operation as Equation 42, the pooled representation $\mathbf{v}$ is computed as:

$$\alpha_t = \frac{\exp\left(\mathbf{q}^\top \mathbf{h}_t\right)}{\sum_{t'} \exp(\mathbf{q}^\top \mathbf{h}_{t'})}, \qquad \mathbf{v} = \sum_t \alpha_t \mathbf{h}_t \tag{42}$$

This formulation is particularly well motivated for LLM-generated text detection, where the most diagnostically informative tokens are not uniformly distributed across a passage. Unusually fluent collocations, atypically smooth transitions, or characteristic high-probability phrases may be concentrated in a small number of positions, and a uniform average dilutes their contribution with that of neutral or uninformative tokens. By allowing the model to learn which positions carry the strongest discriminative signal, attention-based pooling aligns the aggregation mechanism with the structure of the detection task. Within the hybrid architecture developed in this master's thesis, attention-based pooling is applied to the Transformer encoder's output before the pooled neural representation is passed to the fusion layer; the full specification of this component and its interaction with the feature branch is given in Chapter 8.

7. Related Work

Related work on the topic focuses on utilizing various machine learning approaches for the task of binary text classification, primarily divided into traditional linguistic feature-based methods (Schaaff et al., 2023; Gehrmann et al., 2019a) and neural-based fine-tuning methods (Wang et al., 2024a; Yadagiri et al., 2024).

In their work on developing M4GT-Bench, Wang et al. (2024a) fine-tuned multiple Transformer-based models for a binary text classification task, aimed at distinguishing human from LLM-generated text. Their best RoBERTa classifier achieved an accuracy score of 99.3% on non-adversarial data.

From a statistical machine learning perspective, Schaaff et al. (2023) used a set of handcrafted linguistic features combined with traditional classification approaches, including Random Forest (Breiman, 2001), XGBoost (Chen and Guestrin, 2016) and primitive neural-network based Multi-Layer Perceptrons (MLPs) (Haykin, 1994). Their best implementation of a Random Forest classifier resulted in detection accuracies of up to 99.0% for LLM-generated texts, while an XGBoost classifier achieved the highest accuracy of 89.0% for detection of LLM-rephrased texts.

More closely related to the proposed approach, two recent studies (Yadagiri et al., 2024; Knudsen and Hardmeier, 2025) explore hybrid approaches that combine a set of manually extracted linguistic features with Transformer-based embeddings. In both studies features are concatenated with the Transformer output before feeding them into the classification head. The methods differ in that Yadagiri et al. (2024) fine-tune the Transformer together with the classifier, while Knudsen and Hardmeier (2025) keep the Transformer frozen and only train the MLP classification head. Yadagiri et al. (2024)'s hybrid RoBERTa-based architecture achieved an accuracy score of 99.7%. Knudsen and Hardmeier (2025) have similarly reported improved robustness of their hybrid RoBERTa model against evasive adversarial generation techniques compared to Transformer-only baselines. In their experiments, the best hybrid DistilBERT model outperformed baseline DistilBERT (Sanh, 2019) on the control evasive set, with an F1 score of 94.6% as opposed to 86.7%.

Overall, these findings motivate a hybrid architecture on adversarial benchmarks. However, models in existing work have seldom been evaluated on datasets specifically designed to remove formatting artefacts, provide material from multiple LLMs, and include multiple evasion strategies. The RAID benchmark addresses this gap. This master's thesis therefore evaluates the proposed hybrid model directly on RAID to measure its robustness under adversarial perturbations.

8. Method

The overall idea pursued in this master's thesis is to investigate how effective explicit linguistic features are for detecting LLM-generated text, and whether fusing them with a neural Transformer representation yields more robust detection than either signal on its own. This chapter describes the complete architecture and training procedure for the proposed hybrid classifier model designed to examine that question. The chosen approach relies on fusion of hidden representations from a Transformer encoder model with projected attended linguistic features. They contribute to capturing lexical, syntactic, discourse-level, and statistical properties of analysed text. Subsequently, a gated fusion mechanism learns to dynamically weigh these two information streams. The proposed model is built from four main components. These are a RoBERTa encoder that provides contextual representations, a linguistic feature extraction pipeline, a feature attention module with bottleneck projection, and a gating mechanism that fuses these representations before classification.

8.1. Dataset

The data used in this master's thesis is drawn from RealBench (He et al., 2025), a large-scale benchmark for classification between human-written, LLM-generated, and human-LLM collaborative text. The benchmark was released as part of the work on DETree (He et al., 2025) detector. RealBench does not use text generated from scratch. Instead, it aggregates the original samples of several established detection corpora, namely Deepfake (Li et al., 2024), OUTFOX (Koike et al., 2024), TuringBench (Uchendu et al., 2021), M4 (Wang et al., 2024b), and RAID (Dugan et al., 2024). Afterwards, it augments each of them with a common set of hybrid-text attack constructions such as LLM polishing, LLM paraphrasing, perplexity optimization, translation, and extension. Concretely, this master's thesis uses the RAID family within RealBench, so the base texts originate from the RAID benchmark (Dugan et al., 2024), an adversarial dataset specifically designed to evaluate the robustness of LLM-generated text detection, while the attack variants are those constructed on top of the RAID texts by RealBench (He et al., 2025). Unlike simpler datasets where formatting artefacts and stylistic cues provide trivial classification signals, this material includes multiple attack types that actively

attempt to evade detection.

The six attack categories used are shown in Table 1. Each attack type represents a different adversarial strategy, from clean baseline outputs to LLM-based paraphrasing designed to mimic human writing patterns. For readability, the remainder of this master's thesis refers to this material simply as RAID, which throughout should be understood as the RAID subset of RealBench (Dugan et al., 2024; He et al., 2025) rather than the full RAID release.

Table 1.: Attack types used, drawn from the RAID subset of RealBench

Attack Type	**Description**
`no_attack`	Baseline LLM outputs
`synonym`	Word-level synonym substitution
`polish`	Grammar and style polishing
`paraphrase`	Sentence-level restructuring
`perplexity_attack`	Optimized to match human perplexity
`paraphrase_by_llm`	Rewritten by another LLM

For training the proposed model, a balanced selection of approximately 70,000 documents with equal representation across attack types and classes was drawn from the 2,372,075 documents available in the RAID dataset. Here, a document denotes one original RAID text, as opposed to a chunk, which is the fixed-length window later fed to the model. This selection was chosen as a compromise between the computational cost of training and the resulting training effectiveness. Each attack type contributes roughly 11,666 documents, split evenly between the human and LLM classes. The only exception is the `paraphrase_by_llm` attack type, which contains only LLM-paraphrased documents by design. After filtering, the selected corpus has a class distribution of 45.5% human and 54.5% LLM-generated documents.

A common hurdle in fine-tuning encoder-only models for detection of LLM-generated text is insufficient source diversity. When training data comes from a single generative model family, the classification task becomes trivial, as the encoder simply learns superficial stylistic signals specific to that one model rather than actually generalizing to broader language patterns typical to LLM-generated text. To address this limitation, RAID incorporates multiple model families and generations across all attack types, ensuring that learned patterns generalize beyond model-specific artefacts and generation styles. Table 2 shows the top 10 source models, spanning early architectures like GPT-2 (Radford et al., 2019b) to more contemporary models such as GPT-4 (Achiam et al., 2023) and Mistral (Jiang et al., 2023). This diversity in year and family of the models prevents the detector from overfitting to any single model's signals.

Table 2.: Top 10 LLM models by sample amount in final selection

Model	Samples	Year
mistral-chat	2,124	2023
mpt	2,055	2023
llama-chat	2,054	2023
mistral	2,049	2023
gpt2	2,049	2019
mpt-chat	1,970	2023
cohere	1,059	2023
gpt4	1,049	2023
ChatGPT	1,036	2022
gpt3	1,024	2020

RAID's native labels (0=LLM, 1=Human) were inverted for consistency with standard binary classification conventions. Thus, throughout this master's thesis 0 refers to human-written and 1 refers to LLM-generated text. The final selected corpus exhibits natural variation in document length. Human texts average 282.3 tokens (median 226, max 11,789), while LLM outputs average 230.7 tokens (median 226, max 910). In part, this length difference shows the tendency of language models to generate more concise responses, regardless of prompted length requirements.

8.1.1. Data Pre-processing

Several quality filtering steps were applied during dataset pre-processing. Samples containing more than three consecutive repeated words were removed, since such repetition indicates data corruption. Empty or null entries were discarded. Finally, texts shorter than 50 tokens after chunking were filtered out, as these provide insufficient context for reliable classification.

To preserve natural linguistic characteristics and remove possible artefacts, such as lingering HTML tags or non-ASCII characters, a conservative text normalization was applied to the dataset. All text was converted to lowercase to remove capitalization patterns, HTML tags and Markdown formatting were stripped, Unicode characters were normalized to their ASCII equivalents, and runs of multiple whitespace characters were collapsed into a single space. This pre-processing allows the training data to be free of noisy signals that may potentially skew the classifier from learning desired signals that differentiate human from LLM-generated content. Punctuation density, sentence structure, and discourse patterns were deliberately preserved as these may provide valid discriminative signals for text classification.

In order to align with RoBERTa's 512-token limit while processing variable-

length documents, a sliding window strategy with 450-token windows, 350-token stride, and a 50-token minimum threshold was used. This approach results in a 100-token overlap between chunks. Each chunk inherits the label and metadata of its source document. A unique `doc_id` tracks chunks from the same article, enabling document-level aggregation during evaluation. This approach was chosen to reflect real-life usage scenarios where detectors scan documents incrementally, in order to provide a probability score to each section (chunk) of the input. Here, a chunk refers to a single 450-token window and is the unit presented to the model during training and window-level evaluation, whereas a document refers to one original RAID text that may span several chunks. A document shorter than the window limit is treated as exactly one chunk, while a longer one is split into several chunks. Document-level results are obtained by aggregating the chunk predictions that share a `doc_id`, as described in Section 8.3.

Possible information leakage was addressed with grouping chunks by `article_id` before splitting, making sure that all chunks from the same document always stay in the same partition. In addition, a stratified `GroupShuffleSplit` from Scikit-Learn library (Kramer, 2016) was used to maintain balanced attack type representation across splits. The resulting distribution is 80% (54,307 chunks) in the training split, 10%, (6,742 chunks) in the validation split, and 10% (6,806 chunks) in the test split. To ensure a consistent distribution of data, class balance was maintained across all splits. The training set contains 26,815 human chunks and 27,492 LLM-generated chunks. The validation set includes 3,281 human chunks alongside 3,461 LLM chunks. Finally, the test set consists of 3,418 human chunks and 3,388 LLM chunks.

8.1.2. Linguistic Feature Extraction

The final selection of the chosen 25 linguistic features was informed by the statistical feature engineering framework proposed by Schaaff et al. (2023). The selected features are targeted at capturing stylistic differences between human and machine-generated text. These features complement embedded deep representations by capturing additional patterns in vocabulary, syntax, discourse structure, and statistical regularity. Table 3 presents the complete feature set, organized by linguistic level, i.e., by linguistic branch.

Features are computed using SpaCy's `en_core_web_lg` model (Explosion, 2022) for tokenization, POS tagging, and dependency parsing, with NER disabled for processing speed. The GPT-2 model is used to provide perplexity estimates for the analysed texts. Discourse markers are detected using a predefined lexicon of 40 common connectives, e.g., *however*, *therefore*, *for example*. Function words are identified from a list of 70 high-frequency closed-class items, consisting of English articles, prepositions, conjunctions, pronouns.

For normalization, the linguistic features were standardized using Z-score nor-

Table 3.: Final set of 25 linguistic features organized by category

Feature	**Description**	**Category**
msttr	Mean Segmental Type-Token Ratio; lexical diversity	Lexical
avg_word_len	Average character length of words	Lexical
hapax_ratio	Ratio of hapax legomena (words appearing once)	Lexical
function_ratio	Ratio of function words to total words	Lexical
punct_density	Density of punctuation marks	Lexical
char_entropy	Character-level Shannon entropy	Lexical
burstiness	Variance of word recurrence distances	Lexical
repetition_ratio	Ratio of repeated words	Lexical
avg_sent_len	Average sentence length in tokens	Syntactic
sent_len_std	Standard deviation of sentence lengths	Syntactic
noun_ratio	Ratio of nouns to total tokens	Syntactic
verb_ratio	Ratio of verbs to total tokens	Syntactic
adj_ratio	Ratio of adjectives to total tokens	Syntactic
adv_ratio	Ratio of adverbs to total tokens	Syntactic
pron_ratio	Ratio of pronouns to total tokens	Syntactic
pos_diversity	Shannon entropy of POS tag distribution	Syntactic
avg_tree_depth	Average depth of dependency parse trees	Syntactic
max_tree_depth	Maximum depth of dependency parse trees	Syntactic
sub_clause_ratio	Ratio of subordinate clauses	Syntactic
dm_density	Density of discourse markers	Discourse
sent_len_cv	Coefficient of variation of sentence length	Discourse
fp_ratio	Ratio of first-person pronouns (I, me, my, etc.)	Discourse
num_sentences	Total number of sentences	Discourse
words_per_sent	Average words per sentence	Discourse
perplexity	GPT-2 perplexity (sliding window)	LM

malization to ensure that the varying scales of the linguistic metrics, ranging from high-magnitude perplexity scores to lower-frequency discourse marker counts, contribute equally to the model's objective function. This approach prevents features with larger numerical ranges from dominating the weight updates during training and facilitates faster convergence for gradient-based optimization. In order to achieve such normalization of extracted features, a standard scaler is implemented to achieve mean of zero and standard deviation of one. Critically, the standard scaler is fit only on the training set to prevent information leakage, then applied to validation and test sets. This ensures the model cannot exploit distributional information from evaluation data.

8.2. Model Architecture

The proposed pipeline of this master's thesis produces two complementary representations for each input. The first is a 768-dimensional contextual embedding taken from the RoBERTa-base `[CLS]` token, and the second is a 25-dimensional vector of normalized linguistic features. These are processed through separate pathways before being combined via gated fusion. Figure 1 illustrates the complete architecture of proposed hybrid model.

`RoBERTa-base` model is used as the foundational encoder of this master's thesis. The model receives tokenized text chunks of up to 450 tokens, reserving space for special tokens within the 512-token limit. RoBERTa produces contextualized hidden states of shape $(T, 768)$ for input length T. Additionally, a vector $c \in \mathbb{R}^{768}$ is extracted, corresponding to the `[CLS]` position. To reduce overfitting on adversarial data and accelerate training, the bottom six Transformer layers are frozen, while the upper six layers are kept trainable. This approach was chosen to preserve low-level linguistic knowledge of the pre-trained model while allowing adaptation to the classification task.

Not all linguistic features may be equally informative for every sample. Therefore, a learned feature attention mechanism is applied to dynamically re-weight the feature vector (Hu et al., 2018). Given the raw feature vector $\mathbf{x} \in \mathbb{R}^{25}$, attention weights are computed as shown in Equation 43.

$$\mathbf{a} = \text{softmax}\left(W_2 \tanh(W_1 \mathbf{x})\right) \tag{43}$$

Here, $W_1 \in \mathbb{R}^{25\times25}$ is a learned weight matrix that projects the input feature vector into a latent representation, and $W_2 \in \mathbb{R}^{1\times25}$ is a learned vector that maps this latent representation to a scalar compatibility score for each feature. The softmax operation normalizes these scores into attention weights $\mathbf{a} \in \mathbb{R}^{25}$. The attended feature vector is then obtained by element-wise re-weighting, as shown in Equation

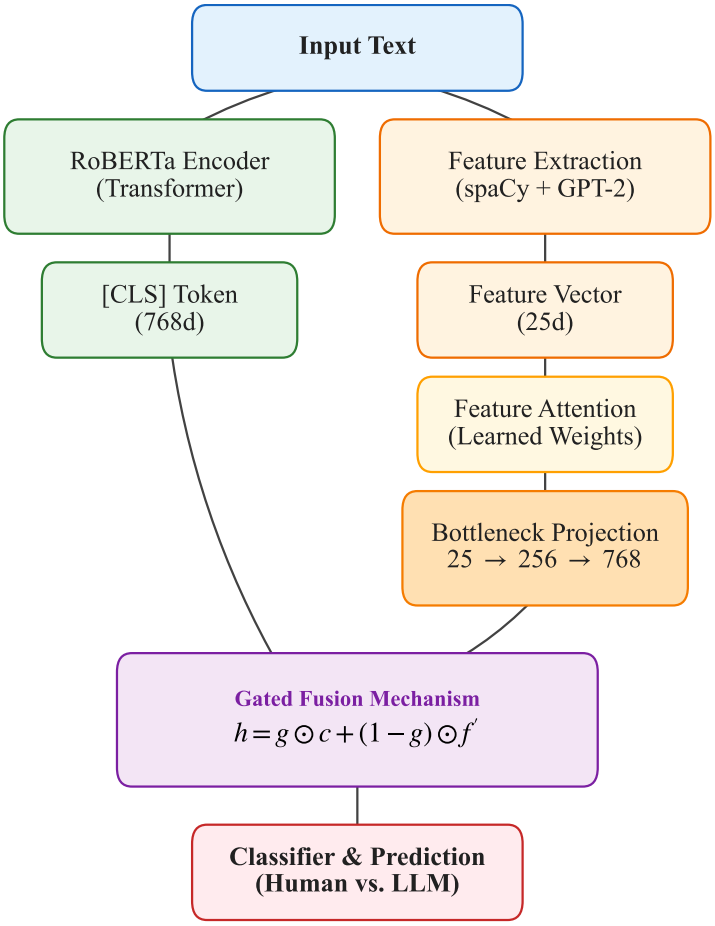

Figure 1.: Architecture of the proposed hybrid RoBERTa model

44.

$$\mathbf{x}_{\text{att}} = \mathbf{a} \odot \mathbf{x} \tag{44}$$

This mechanism enables the model to emphasize highly discriminative features, while down-weighting less informative ones on a per-sample basis.

The linguistic features are projected through a two-stage bottleneck (Nagrani et al., 2021) with dimension expansion, in order to align them with the semantic space of the RoBERTa embedding, using GELU as shown in Equations 45 and 46.

$$f_{\text{mid}} = \text{GELU}(\text{LayerNorm}(W_{25\to256} f_{\text{att}})) \tag{45}$$

$$f'' = \text{GELU}(\text{LayerNorm}(W_{256\to768} f_{\text{mid}})) \tag{46}$$

Dropout of $p = 0.15$ is applied after each projection. The bottleneck design of 25

$\rightarrow 256 \rightarrow 768$ allows the model to learn feature interactions in the intermediate 256-dimensional space before final expansion.

The core architectural contribution of the proposed hybrid model is a learned gating function that determines the relative importance of both information sources during classification (Chaudhari et al., 2021; Hu et al., 2018). The gate value $g \in [0,1]^{768}$ is computed from the concatenation of input representations, as shown in Equation 47, where $W_g \in \mathbb{R}^{768\times1536}$ and σ denotes the sigmoid function.

$$g = \sigma(W_g[c\,;\, f'']) \tag{47}$$

The fused representation can be presented as the element-wise convex combination shown in Equation 48.

$$h = g \odot c + (1 - g) \odot f'' \tag{48}$$

Specifically, gate values near 1 indicate reliance on RoBERTa's understanding from hidden representations, while values near 0 emphasize the reliance on linguistic features. This adaptive mechanism allows the model to dynamically adjust its classification strategy based on the input.

At the final stage of classification, a fused vector h passes through a two-layer MLP classifier (Haykin, 1994) with the structure outlined in Equations 49 and 50.

$$h_1 = \text{GELU}(\text{LayerNorm}(W_{768\rightarrow768}h)) \tag{49}$$

$$\hat{y} = W_{768\rightarrow2}h_1 \tag{50}$$

The classifier head produces final logits for the binary classification task of human written as opposed to LLM-generated text.

8.3. Evaluation

All models are evaluated at two scopes to reflect both technical performance and practical deployment scenarios, namely at the window level and the document level.

For window-level evaluation, each 450-token sliding window chunk is treated as an independent sample. The standard classification metrics are then computed for it, namely accuracy, precision, recall, macro-averaged F1 score, and AUC-ROC. This evaluation scope reflects per-chunk performance and is used for monitoring during training.

For document-level evaluation, predictions from all associated chunks identified by `doc_id` are collected. The aggregation uses mean probability to give the final metric, which means given n chunks from document d, each producing predicted probability p_i for the respective class, the document-level probability is defined in Equation 51.

$$P_d(\text{class}) = \frac{1}{n}\sum_{i=1}^{n} p_i \tag{51}$$

The final document label is finally determined by thresholding the regressive value at 0.5. Document-level metrics better reflect real-world deployment where entire articles or passages must be classified, and are used as the primary evaluation criterion in this master's thesis. Performance is evaluated at both window and document levels using accuracy, macro-averaged precision, recall, and F1, accompanied by AUC-ROC and confusion matrices.

Since one of the central research questions of this master's thesis concerns robustness under adversarial conditions rather than average accuracy, document-level performance is additionally reported separately for each RAID attack type. The test predictions are grouped by their attack label and the metrics are recomputed within each group, which requires no retraining because it only re-partitions predictions that the models already produce. The drop in macro F1 relative to the clean `no_attack` subset is then used as a direct measure of degradation under each attack. To establish whether the differences between the models are reliable rather than an artefact of the particular test split, McNemar's exact test is applied to the paired document-level decisions of the compared models (Dietterich, 1998), and a paired bootstrap with 10,000 resamples is used to place a confidence interval on the macro F1 difference.

9. Experimental Setting

This chapter outlines the pipeline's training procedure, baseline models and their configuration, evaluation protocols, and implementation details necessary for reproducing the results.

9.1. Training Configuration

The models in this master's thesis are trained with cross-entropy loss, incorporating two modifications for improved generalization. First, class weights are computed using an inverse frequency approach to account for the slight label imbalance, i.e., 45.5% Human samples as opposed to 54.5% LLM-generated samples. Second, label smoothing with $\epsilon = 0.05$ is used to soften hard targets, which helps to prevent overconfident predictions.

The optimizer of choice for this master's thesis is AdamW (Loshchilov and Hutter, 2019), used with two separate learning rates for the hybrid model's components, namely 2×10^{-5} for RoBERTa layers and 1×10^{-4} for the feature attention, projection, gating, and classifier head components. Weight decay for fine-tuning is set to 0.02. Gradients are clipped to maximum norm of 1.0 to prevent instability. A ReduceLROnPlateau scheduler is used with patience=4, factor=0.5, minimum LR=1×10^{-7}, in order to reduce learning rates when validation document-level F1 score plateaus during late epochs of the fine-tuning process.

Table 4.: Complete hyperparameter configuration for the hybrid model

Hyperparameter	**Value**
Training	
Batch size	16
Maximum epochs	25
Early stopping patience	8
Mixed precision (FP16)	Yes
Optimization	
Optimizer	AdamW
LR (RoBERTa layers)	2×10^{-5}
LR (Head/Fusion layers)	1×10^{-4}
Weight decay	0.02
Gradient clipping	max_norm=1.0
LR scheduler	ReduceLROnPlateau
Patience	4
Factor	0.5
Minimum LR	1×10^{-7}
Regularization	
Dropout	0.15
Label smoothing	0.05
Class weights	Inverse frequency
Reproducibility	
Random seed	42

A complete list of hyperparameters used to fine-tune the proposed hybrid classifier model are displayed in Table 4. These were inspired by standard training hyperparameter ranges for fine-tuning encoder-only models on a classification task. Although training is set to proceed for up to 25 epochs with early stopping, the best-performing checkpoint based on validation document-level F1 score is selected for final evaluation on the test set. Namely, Epoch 14. For fair baseline comparison, the RoBERTa-only model is evaluated at the same epoch as the best hybrid model. All experiments use random seed 42 for reproducibility across PyTorch, NumPy, and CUDA.

9.2. Baseline and Ablation Study

Two baselines were selected in this master's thesis in order to evaluate the contribution of the proposed hybrid model's classification performance. Both of them are outlined in this section. Baseline A uses the same `roberta-base` encoder with an identical classification head architecture. The baseline choice is to fully exclude the linguistic feature fusion and gating mechanism. The model receives only tokenized

text as input and relies purely on learned hidden representations. To ensure a fair comparison, the baseline model is trained with the exact same hyperparameters, batch size, optimization settings, and regularization as the hybrid model. It is crucial to mention that the checkpoint from Epoch 14 is used for final evaluation, matching the training duration of the best-performing hybrid model. This choice isolates the contribution of explicit linguistic features by controlling for all other architectural and training factors, as well as prevents the baseline model from potentially learning longer during additional epochs.

Baseline B evaluates the discriminative power of linguistic features in isolation. The architecture is a simple feed-forward MLP network, similar to that of the classifier head used in the proposed hybrid model. Since this model processes only feature vectors rather than text, as well as having a much more trivial architecture, a different set of training settings optimized for the smaller model had to be used, namely a batch size of 256, 50 training epochs, the Adam optimizer with a learning rate of 1×10^{-3}, and standard cross-entropy loss with class weights.

In addition to these two baselines, three ablated variants of the proposed hybrid model are trained in order to isolate the contribution of the gating mechanism itself, rather than the contribution of the linguistic features in general. Each variant is trained with identical hyperparameters, namely the same seed, optimizer, learning rates, regularization, and epoch budget, and differs only in how the contextual and linguistic representations are combined, i.e., in the approach for the gate. The concatenation variant replaces the gate with a plain concatenation of the two vectors before the classifier, matching the fixed combination strategy used by Knudsen and Hardmeier (2025) discussed in Chapter 7. The additive variant sums the two vectors without a gate. The third variant retains the gate but removes the per-feature attention module. Comparing these variants against the proposed hybrid model separates the effect of the gate and of the feature attention from the effect of just adding linguistic features to the encoder.

10. Results

This chapter examines the experimental results. It first establishes the overall performance of the proposed hybrid model against the two single-modality baselines, then examines robustness across the individual RAID attack types, the structure of the remaining errors, the behaviour of the gating mechanism, the relative contribution of the linguistic features, and finally a controlled ablation that isolates the gated fusion from the other components.

10.1. Overall Model Performance

Table 5 presents the evaluation of the proposed hybrid model against two baselines on the test set. The metrics are aggregated at both the window level over individual 450-token chunks and the document level per source unit.

The hybrid model achieves a document-level F1 score of 95.05%, outperforming the RoBERTa-only baseline by 3.47 percentage points and the feature-only baseline by 10.46 points. At the document level, the model shows balanced precision and recall, namely 95.36% precision against 94.88% recall, together with an AUC-ROC of 97.72%. This balance indicates that the model does not systematically favour one class over the other despite the slight class imbalance in the dataset, and the high AUC suggests reliable ranking of confidence scores, which is useful in real-world deployment scenarios where a decision threshold must later be tuned. The feature-only baseline, while achieving respectable performance at 84.59% F1, confirms that the linguistic features carry genuine discriminative power on their own but benefit substantially from fusion with deep Transformer representations.

In order to check if the improvement over the Transformer baseline is indeed reliable, rather than an artefact of the particular test split, a paired McNemar test and a bootstrap confidence interval were computed on the document-level decisions. The hybrid model classifies 318 documents correctly that the RoBERTa baseline misclassifies while falsely classifying only 106. This is a ratio of roughly three to one, and the exact McNemar two-sided p-value is below 10^{-12}. The paired bootstrap places the macro F1 advantage at +0.0347 with a 95% confidence interval of [+0.0283, +0.0411], which excludes zero. The comparison against the feature-only baseline gives a larger and equally clear gap.

Model	F1	Acc	AUC
Win-Level			
Hybrid	**95.02**	**95.02**	97.72
RoBERTa	91.29	91.32	**98.24**
MLP Classifier	84.78	84.79	92.13
Doc-Level			
Hybrid	**95.05**	**95.07**	97.72
RoBERTa	91.58	91.70	**98.24**
MLP Classifier	84.59	84.62	92.13

Table 5.: Performance comparison on RAID test set

		Macro-F1			LLM detection rate (TPR)		
Attack	H/L	Hybrid	RoBERTa	Feat.	Hybrid	RoBERTa	Feat.
No attack (clean)	569/565	**97.44**	94.26	91.09	97.35	**98.23**	91.68
Synonym	602/556	**97.83**	97.75	89.79	96.04	**97.66**	82.91
Polish	596/546	**89.58**	79.80	76.03	91.94	**95.05**	84.98
Paraphrase	586/578	**92.61**	86.06	81.61	93.25	**95.67**	81.31
Perplexity attack	577/567	97.11	**97.20**	80.67	96.65	**98.24**	65.08
Paraphrase by LLM	0/550	n/a	n/a	n/a	96.73	**98.18**	91.27
All (pooled)	2930/3362	**95.05**	91.58	84.59	95.33	**97.17**	82.81

Table 6.: Document-level robustness by RAID attack type

10.2. Robustness Across Attack Types

The headline comparison reports a single pooled score, but the central research question concerns robustness under adversarial conditions, and the six RAID attacks differ widely in how hard they are to detect. In order to examine robustness rather than average accuracy, the document-level predictions were grouped by attack type and the metrics recomputed within each group. Table 6 reports the macro F1 and the LLM detection rate for each attack, alongside the number of human and LLM documents in each subset.

Several patterns emerge from the breakdown. On the clean subset and on the two milder attacks, namely synonym substitution and the perplexity attack, all models perform well, and the hybrid is level with or slightly ahead of the Transformer baseline. The differences appear on the two hardest attacks, namely grammar and style polishing and sentence-level paraphrasing. Here, the hybrid model retains a macro F1 of 89.58 and 92.61 while the RoBERTa baseline falls to 79.80 and 86.06.

The hybrid advantage is therefore largest exactly where detection is most difficult, with a margin of about ten points on polishing and seven points on paraphrasing.

Robustness can be read more directly as the loss in performance relative to the clean subset. The RoBERTa baseline loses 14.46 macro F1 points under polishing and 8.20 under paraphrasing. The hybrid model loses only 7.87 and 4.83 points on the same attacks. The combination is thus more robust to these perturbations than the Transformer alone, which is the behaviour anticipated when the two information sources respond differently to adversarial manipulations. The `paraphrase_by_llm` subset, which contains only machine-generated documents, does not get a two-class F1, but the detection rate there stays high for every model, around 96.73 for the hybrid, so this attack is not the hardest condition despite rewriting the text with a second model.

The mechanism behind the difference is visible in the class-conditional rates. Under polishing and paraphrasing, the RoBERTa baseline keeps a high detection rate on LLM-generated text, yet its score on human text collapses to 66.28 and 76.79, meaning it increasingly mislabels adversarially edited human writing as LLM-generated. The linguistic features restore much of this specificity in the hybrid model, raising it to 87.42 and 91.98 on the same subsets. The gain in robustness therefore comes mainly from fewer false alarms on attacked human text rather than from catching more LLM-generated text. This evidence supports the assumption that combining the two sources of representation, namely linguistic features and deep representations, is more resilient than relying on either alone.

10.3. Performance Across LLMs

The RAID subset used here draws its machine-generated text from twelve LLMs spanning several families and release years, and a detector that performs well on average may still fail on particular families. Because the human documents in the test set are not attributable to any LLM, per-LLM performance is shown as the detection rate, namely the proportion of a given model's documents that are correctly identified as LLM-generated. Table 7 reports this rate for each LLM, with the pooled rate over all LLMs in the final row.

The detection rate is high and fairly uniform across LLMs for both the hybrid model and the RoBERTa baseline, staying above 93% for eleven of the twelve models. Detectability does not decrease at the same rate with the LLMs age, since the early GPT-2 is detected as reliably as the more modern GPT-4. The chat-tuned variants are detected at rates similar to or slightly above their base counterparts. Cohere is the clear outlier, with the hybrid model detecting only 78.26% of its documents against 91.30% for RoBERTa, and the feature-only baseline collapsing to 55.07%. The RoBERTa baseline scores a marginally higher detection rate compared

LLM	n	Hybrid	RoBERTa	Ling. Feat. Only
GPT-2	198	**98.48**	**98.48**	75.76
GPT-2 XL	202	94.06	**97.03**	79.21
GPT-3	183	97.27	**97.81**	83.61
ChatGPT	220	99.55	**100.00**	88.64
GPT-4	172	97.67	**99.42**	84.30
Cohere	207	78.26	**91.30**	55.07
Cohere-Chat	208	93.75	**95.19**	75.48
LLaMA-Chat	429	98.37	**99.30**	89.98
Mistral	390	90.77	**93.08**	78.21
Mistral-Chat	391	98.72	**99.23**	88.49
MPT	351	94.87	**95.73**	83.76
MPT-Chat	411	98.05	**98.78**	92.21
All models	3362	95.33	**97.17**	82.81

Table 7.: Document-level detection rate by LLM, reported as a percentage

to the hybrid model on almost every LLM, typically by one to three points. This is the expected counterpart of the mechanism identified earlier, since the hybrid model trades a small amount of recall on machine-generated text for a large reduction in false positives on human text, raising correctly detected human text from 85.43% to 94.78%. Because that gain applies to every LLM equally, the small per-LLM recall difference may not overturn the hybrid model's overall advantage, and this explains why the hybrid model leads on balanced macro-averaged metrics despite the baseline model's slightly higher raw detection rate on individual LLMs.

10.4. Error Analysis

At the document level, the hybrid model produces 153 false positives against 157 false negatives on the test set, which corresponds to a false positive rate of 5.2% and a false negative rate of 4.7%. The two error types are therefore close to balanced overall. This stands in clear contrast to the RoBERTa-only baseline, which produces 427 false positives against 95 false negatives on the same documents, corresponding to a false positive rate of 14.6%. The largest single effect of adding the linguistic features is thus the suppression of false alarms on human text, achieved while accepting a small rise in missed LLM-generated text from 2.8% to 4.7%.

The remaining errors are not spread evenly across conditions. As the per-attack breakdown in Table 6 implies, most of the hybrid model's false positives happen on the polishing and paraphrasing subsets, with 75 and 47 false positives respectively.

Subset	Mean Gate Value (g)
All Samples	0.4390
Human Samples	0.3966
LLM Samples	0.4817

Table 8.: Learned gate values for test set samples

This concentration is expected, since these attacks rewrite human text in ways that introduce the kind of surface uniformity the model associates with LLM-generated text. The false negatives are similarly concentrated on the harder attacks, which rewrite LLM-generated text toward more human-like text.

10.5. Gating Mechanism Analysis

The gated fusion mechanism proposed for the hybrid model provides interpretable insights into how the model balances two sources of information in its decision-making. Table 8 summarizes the learned gating behaviour across the test set.

The average gate value of 0.4390 indicates a slight overall preference for linguistic features over RoBERTas hidden representations. The class-conditional gating behaviour is more revealing. For human-written text, the model assigns lower gate values ($g = 0.3966$), placing greater weight on linguistic features. In contrast, the model learns a more balanced fusion ($g = 0.4817$) for LLM-generated texts. This pattern suggests that human writing exhibits more distinctive and reliable stylistic signals captured by the extracted features, while LLM-generated text requires using both global context and stylistic cues for accurate classification. The learned gating behaviour is consistent with the proposed architectural choice, since it lets the classifier apply dynamic, sample-specific fusion instead of a fixed combination. Whether this dynamic fusion actually outperforms a fixed one is examined directly in the ablation study in Section 10.7.

10.6. Feature Importance and Attention

The second research question asks which linguistic features contribute most to detection. To answer this, two complimentary experiments are reported. The first is the learned attention weight inside the hybrid model, which shows where the model allocates weight across the 25 linguistic features. The second is permutation importance, a direct and model-agnostic measure of how much a feature contributes to accuracy. It is obtained by randomly shuffling the values of a single feature across the test samples, which destroys the association between that feature and the

Feature	Family	ΔF1 (perm.)	Attn. rank
perplexity	language-model	0.163	22
num_sentences	syntactic	0.074	17
hapax_ratio	lexical	0.073	8
avg_word_len	lexical	0.045	25
pos_diversity	syntactic	0.038	13
pron_ratio	syntactic	0.037	16
msttr	lexical	0.035	2
punct_density	lexical	0.026	3
burstiness	lexical	0.025	24
sent_len_cv	syntactic	0.019	5

Table 9.: Top ten features by permutation importance, with each feature's attention rank

label while leaving every other feature untouched, and then measuring how far the document-level macro F1 falls. A large drop means the model genuinely relies on that feature for its decisions, whereas a negligible drop means the feature could be removed with little effect to models performance. This measure is important here because attention weights only reveal internal allocation and could also be high for a feature that barely moves the final prediction, while permutation importance answers the actual research question, namely how much each feature is worth for the decision itself. The values reported below are averaged over 25 shuffles per feature on the feature-only model. Table 9 lists the ten most important features together with their rank under the attention mechanism.

Perplexity is the single most important feature by a clear margin, with a macro F1 drop of 0.163 when permuted, roughly twice that of the next most important feature. It is followed by structural and lexical diversity measures, namely the number of sentences, the hapax ratio, average word length, and POS diversity. The lexical and syntactic families contribute the most, if looked at by linguistic family, with summed importance of 0.252 and 0.250. The single perplexity feature contributes 0.163, and the discourse features contribute little at 0.015. This ordering is consistent with the view that the strongest stylometric signals of machine generation lie in predictability and in lexical and structural variety rather than in discourse markers.

Interestingly, these two views diverge in an informative way. Perplexity ranks first for contribution to accuracy but only twenty-second by attention weight, while the repetition ratio, which the attention mechanism ranks first, ranks only thirteenth by importance. The attention weights therefore behave as a soft saliency signal rather than a direct measure of contribution, and the two should be read together rather than as interchangeable signals of feature importance. This distinction matters

Model	Acc	Macro-F1	AUC	LLM TPR
Hybrid (gated fusion), full model	**95.07**	**95.05**	97.72	95.33
Concatenation fusion	94.52	94.48	**98.73**	96.34
Additive fusion	93.55	93.53	98.08	92.98
Gated, no feature-attention	93.96	93.90	95.89	96.91
RoBERTa-only	91.70	91.58	98.24	**97.17**
Feature-only MLP	84.62	84.59	92.13	82.81

Table 10.: Fusion ablation study at the document level

for the interpretability claim of this master's thesis, since the learned attention alone would have overstated the role of repetition and understated the role of perplexity. The features that the attention mechanism favours most, namely the repetition ratio, the mean segmental type-token ratio, and punctuation density, remain reasonable stylistic indicators, but they are best treated as a secondary and qualitative view alongside the permutation results.

The robustness research question can be addressed by repeating the importance computation within each attack type. Perplexity remains the top feature under every attack, yet its importance is partly reduced by exactly the attacks that target surface statistics, falling from 0.196 on clean text to 0.113 under the perplexity attack and 0.120 under polishing attack, while the lexical and structural features stay comparatively stable throughout. This is consistent with the broader finding that no single signal is robust on its own and that the fused model benefits from drawing information from several signals at once. The mean gate value of 0.439 reported above, which leans slightly toward the linguistic branch, fits this picture, since the features carry genuine signal that the model is willing to rely on for its decision-making.

10.7. Ablation Study

As discussed in Chapter 9, comparing the hybrid model against the two single-modality baselines tests only whether the linguistic features help at all, not whether the gate is responsible for the gain. To isolate the gate, the three ablated variants described in Chapter 9 were trained. Table 10 reports the document-level results.

All four fusion variants outperform the RoBERTa-only baseline, which confirms that the linguistic features help regardless of how they are combined. Among the fusion strategies, the gated model attains the highest macro F1 and accuracy at the decision threshold. The margin over plain concatenation is small at 0.57 macro F1 points, but it is consistent and statistically detectable, with a McNemar

exact p-value of 0.037. The margins over additive fusion and over the gate without feature attention are larger, at 1.52 and 1.15 macro F1 points respectively, both clearly significant. Removing the feature attention therefore costs about as much as replacing the gate with a simpler combination, which justifies retaining that component as well. These margins are estimated from a single training run of each variant at a fixed random seed, so the McNemar test quantifies variation over the test documents but not over training initializations. The smallest of the differences, the 0.57-point macro F1 margin over concatenation, should be read with corresponding caution, since a gap of this size could plausibly move under reseeding, whereas the larger margins over additive fusion and over the variant without feature attention are less sensitive to this source of variance.

As with the overall comparison, the advantage of the gate is concentrated on the hardest attacks. On clean and easy attacks, the variants are within noise levels of one another, and concatenation is in fact marginally ahead on the clean subset. In contrast, the gated model reaches 89.58 macro F1 on the polishing attack against 86.85 for concatenation, 86.25 for additive fusion, and 85.42 without attention, with a similar score distribution on paraphrasing attack. The benefit of the gate is therefore real but modest, and it is most useful precisely under the conditions the benchmark was built to stress.

One caveat deserves mention. The concatenation variant attains a higher AUC of 98.73 than the gated model despite its lower F1 at the 0.5 threshold, which indicates that its ranking of documents is good while its decisions are less well calibrated at this particular operating point. The evidence for the gate should accordingly be read as support for better calibrated decisions on difficult inputs rather than for stronger class separation.

11. Discussion

The experimental results indicate that traditional linguistic features, when fused with Transformer embeddings through a gated fusion mechanism, provide measurable improvements for adversarial LLM-generated text detection. The gain of 3.47 percentage points in document-level F1 over a strong RoBERTa baseline is consistent across resampling and statistically significant. It is a moderate rather than a dramatic improvement, but it is obtained on the challenging RAID benchmark, where the attacks are specifically designed to evade detection systems.

The score improvement of the proposed hybrid architecture can be attributed to the complementary nature of both linguistic features and deep representations. While RoBERTa captures global contextual patterns and semantic coherence, the manually extracted linguistic features encode explicit knowledge about syntactic complexity, lexical diversity, and discourse structure that Transformers may only partially or incompletely learn. The proposed gated fusion mechanism enables the model to dynamically choose between these two sources of information for the classification.

The feature analysis shows systematic differences between human and LLM-generated text that explain the hybrid model's strong performance. Human writing displays higher variability in sentence length, richer part-of-speech diversity, and higher burstiness in how words are used. These properties are similar to properties of natural language. In contrast, LLM-generated text is much more uniform. Its lower perplexity and reduced variation in sentence length indicate a trend toward uniform language and predictable phrasing. This may be a consequence of how LLMs are trained, since next-token prediction objectives prioritize high-probability output over stylistic variation. LLM output also contains slightly longer words on average, likely reflecting a bias toward more formal vocabulary shaped by the composition of pretraining corpora, which tend to over-represent edited and formal writing.

The per-attack analysis gives a more detailed picture of where the improvement comes from. Across the six RAID attacks the hybrid model degrades least on the two hardest conditions, namely style polishing and sentence-level paraphrasing, where the RoBERTa baseline loses considerably more performance. The reason is visible in the error structure. At the document level, the hybrid model makes a near balanced number of false positives and false negatives. Under the hardest attacks the baseline increasingly flags adversarially edited human text as LLM-generated,

and the linguistic features restore much of the lost specificity, which is the main source of the hybrid model's gain. Interestingly, the `paraphrase_by_llm` attack is not the hardest condition, since detection rates on it stay high for every model. The hardest conditions are instead the polishing and paraphrasing attacks that rewrite human text toward LLM-like uniformity, which is also where the remaining false positives concentrate.

The gated fusion mechanism may be seen as the main contribution of this master's thesis. Unlike simple concatenation or additive combination strategies seen in prior work, the learned gate allows the model to dynamically adjust the influence of RoBERTa and linguistic features based on each sample. The class-conditional gating behaviour (human $g = 0.40$, LLM $g = 0.48$) demonstrates that the model learns interpretable fusion strategies rather than arbitrary combinations. The fusion ablation supports this claim, since the gated model outperforms plain concatenation, additive fusion, and a variant without feature attention, although the margin over concatenation is small and is clearest on the hardest attacks. The contribution of the gate is therefore better described as a modest and well-targeted improvement in the calibration of decisions on difficult inputs, rather than as a large gain in overall model performance.

The feature attention mechanism similarly contributes to model interpretability by identifying which linguistic features the model attends to most for the tested data. The high attention weights assigned to repetition ratio, lexical diversity (MSTTR), and syntactic complexity (tree depth) are consistent with linguistic intuitions about human and LLM-generated texts. A complementary permutation analysis qualifies this interpretation. When the features are ranked by their measured contribution to accuracy, rather than by attention weight, perplexity becomes the dominant feature, even though the attention mechanism ranks it low. This implies that attention weights and their contribution to accuracy are related, but not in the same quantity. The interpretability claims of this master's thesis therefore rest more safely on the permutation results, with the attention weights serving as a secondary and qualitative view.

A central caveat concerns the external validity of these results. The reported accuracy is high, but it is accuracy under matched conditions, measured on held-out data drawn from the same distribution as the training data. Detecting machine-generated text is, in essence, the separation of human writing from the characteristic statistical and stylistic signatures of particular LLMs. Therefore, the classifier learns precisely the signatures that are present in its training distribution. The RAID dataset is deliberately broad, spanning several model families and generations together with six attack types, which makes these signatures more varied than in single-source corpora and reduces the risk of the model learning the stylistic signature of any single LLM. Even so, RAID is a sample of the space of

possible models rather than a complete collection of it. Models, decoding strategies, prompting schemes, fine-tuning approaches, and writing domains that lie outside its coverage may carry signatures the classifier has never observed, and there is no guarantee that the learned decision boundary transfers to them. The numbers reported here should therefore be read as evidence of what the approach can achieve within the distribution it was trained on, not as a claim of universal detection capability.

A closely related caveat concerns durability over time. The signal the detector relies on most strongly is also the one most likely to erode and change with time. The permutation analysis identifies perplexity as the dominant feature, yet low perplexity is exactly the property that newer and better-aligned language models may reduce, as their output becomes more fluent and more human-like. This is also the property that the perplexity-targeting attack is built to neutralize. As generation methods continue to change, the distributional gap between human and machine text that the model exploits may be expected to narrow, a form of concept drift that affects supervised detectors in general (Wu et al., 2025; He et al., 2024b). The detector is therefore not future-proof. Sustained deployment would require periodic retraining on text from current generators and, most likely, recalibration for each target domain.

Several narrower limitations also apply. The set of 25 linguistic features balances coverage against computational cost, so a larger or differently chosen set might capture additional patterns. In addition, some of the current features may weaken as language models evolve. The sliding window over 450-token chunks, although necessary given the encoder's context limit, may break long-range patterns that a document-level architecture could otherwise exploit. The reported figures also come from a single training run per model at a fixed seed, so the comparisons are not accompanied by an estimate of run-to-run variance from random initialization and data ordering. The larger gaps reported in this master's thesis are unlikely to be explained by such variance, but the small margins between the fusion variants would need repeated runs with different seeds to be confirmed for sure. Finally, the evaluation assumes that whole texts are classified, whereas an adversary who edits only selected passages or combines several generation strategies could make detection considerably harder, and the RAID attack types cover only part of this adversarial attack space.

12. Conclusion

This master's thesis aimed at investigating whether explicit linguistic features, when combined with Transformer-based embeddings, improve the binary classification of written text as human-written or LLM-generated. Using the adversarial RAID dataset, the proposed hybrid architecture with gated fusion demonstrates that extracted linguistic signals provide measurable positive value beyond what modern Transformers learn implicitly.

The supporting experiments show more specific tendencies relevant to the two research questions. Broken down by attack type, the hybrid model is most clearly ahead on the hardest adversarial conditions, namely style polishing and paraphrasing, where it suppresses the false positives that the Transformer baseline produces on edited human text. This addresses the question of robustness. Next, a permutation analysis identifies perplexity as the single most influential linguistic feature, followed by structural and lexical diversity measures, which addresses the question of feature contribution. A controlled ablation shows that the gated fusion outperforms fixed concatenation and additive combination, although the margin is modest and concentrated on difficult inputs. In addition, a paired significance test confirms that the overall improvement over the Transformer baseline is unlikely to be due to randomness. These answers hold within the limits of a single adversarial benchmark and a fixed family of linguistic features, and they describe the generators and attacks represented in RAID rather than all LLM-generated text in general.

As LLMs continue to evolve, hybrid approaches that combine the representational power of large-scale pretraining with the interpretability of manually extracted features may prove important for robust detection systems. By demonstrating that linguistic features provide additive value when properly integrated with Transformer embeddings, this work suggests a direction for future work of developing detection systems that can adapt to evolving generation strategies while maintaining transparency in their decision-making processes. Future work may explore extending this hybrid framework to multi-class and cross-domain detection settings, as well as evaluating its robustness against emerging paraphrasing and obfuscation strategies.

Bibliography

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. (2023). GPT-4 technical report. *CoRR*, abs/2303.08774.

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang (2018). Generating natural language adversarial examples. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2890–2896.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton (2016). Layer normalization. *CoRR*, abs/1607.06450.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio (2014). Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency (2018). Multimodal machine learning: A survey and taxonomy. *IEEE transactions on pattern analysis and machine intelligence*, 41(2):423–443.

Regina Barzilay and Mirella Lapata (2008). Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34(1):1–34.

Yonatan Belinkov and Yonatan Bisk (2018). Synthetic and natural noise both break neural machine translation. In *6th International Conference on Learning Representations (ICLR)*.

Christopher M Bishop and Nasser M Nasrabadi (2006). *Pattern recognition and machine learning*, volume 4. Springer, New York, NY.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Nicholas Boucher, Ilia Shumailov, Ross Anderson, and Nicolas Papernot (2022). Bad characters: Imperceptible NLP attacks. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1987–2004. IEEE.

Leo Breiman (1996). Bagging predictors. *Machine learning*, 24(2):123–140.

Leo Breiman (2001). Random forests. *Machine Learning*, 45(1):5–32.

Kay Henning Brodersen, Cheng Soon Ong, Klaas Enno Stephan, and Joachim M Buhmann (2010). The balanced accuracy and its posterior distribution. In *2010 20th international conference on pattern recognition*, pages 3121–3124. IEEE.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

C2PA (2023). Content provenance and authenticity explainer. Technical report, C2PA. Version 1.3.

Sneha Chaudhari, Varun Mithal, Gungor Polatkan, and Rohan Ramanath (2021). An attentive survey of attention models. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 12(5):1–32.

Tianqi Chen and Carlos Guestrin (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794.

Kyunghyun Cho, Bart Van Merriënboer, Çağlar Gulçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio (2014). Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1724–1734.

Evan N. Crothers, Nathalie Japkowicz, and Herna L. Viktor (2023). Machine-generated text: A comprehensive survey of threat models and detection methods. *IEEE Access*, 11:70977–71002.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier (2017). Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 933–941. JMLR.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186.

Thomas G. Dietterich (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1923.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, Jing Yi, Weilin Zhao, Xiaozhi Wang, Zhiyuan Liu, Hai-Tao Zheng, Jianfei Chen, Yang Liu, Jie Tang, Juanzi Li, and Maosong Sun (2023). Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5(3):220–235.

Carl Doersch, Abhinav Gupta, and Alexei A. Efros (2015). Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1422–1430.

Liam Dugan, Alyssa Hwang, Filip Trhlík, Andrew Zhu, Josh Magnus Ludan, Hainiu Xu, Daphne Ippolito, and Chris Callison-Burch (2024). RAID: A shared benchmark for robust evaluation of machine-generated text detectors. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12463–12492, Bangkok, Thailand. Association for Computational Linguistics.

Liam Dugan, Andrew Zhu, Firoj Alam, Preslav Nakov, Marianna Apidianaki, and Chris Callison-Burch (2025). GenAI content detection task 3: Cross-domain machine generated text detection challenge. In *Proceedings of the 1stWorkshop on GenAI Content Detection (GenAIDetect)*, pages 377–388, Abu Dhabi, UAE. International Conference on Computational Linguistics.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou (2018). Hotflip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 31–36.

Steffen Eger, Gözde Gül Şahin, Andreas Rücklé, Ji-Ung Lee, Claudia Schulz, Mohsen Mesgar, Krishnkant Swarnkar, Edwin Simpson, and Iryna Gurevych (2019). Text processing like humans do: Visually attacking and shielding NLP systems. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1634–1647, Minneapolis, Minnesota. Association for Computational Linguistics.

Charles Elkan (2001). The foundations of cost-sensitive learning. In *International joint conference on artificial intelligence*, volume 17, pages 973–978. Lawrence Erlbaum Associates Ltd.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 226–231.

Explosion (2022). spacy: Industrial-strength natural language processing in python (with `en_core_web_lg` model). `https://spacy.io`. Version 3.x, large English pipeline `en_core_web_lg`.

Sarah Farhadpour, Timothy A Warner, and Aaron E Maxwell (2024). Selecting and interpreting multiclass loss and accuracy assessment metrics for classifications with class imbalance: Guidance and best practices. *Remote Sensing*, 16(3):533.

Tom Fawcett (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874.

Christian Federmann, Oussama Elachqar, and Chris Quirk (2019). Multilingual whispers: Generating paraphrases with translation. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 17–26, Hong Kong, China. Association for Computational Linguistics.

Yoav Freund and Robert E Schapire (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139.

Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi (2018). Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 50–56. IEEE.

Sebastian Gehrmann, Hendrik Strobelt, and Alexander Rush (2019a). GLTR: Statistical detection and visualization of generated text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 111–116, Florence, Italy. Association for Computational Linguistics.

Sebastian Gehrmann, Hendrik Strobelt, and Alexander M Rush (2019b). Gltr: Statistical detection and visualization of generated text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 111–116.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. MIT Press.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy (2015). Explaining and harnessing adversarial examples. In *3rd International Conference on Learning Representations (ICLR)*.

Shreya Goyal, Sumanth Doddapaneni, Mitesh M Khapra, and Balaraman Ravindran (2023). A survey of adversarial defenses and robustness in nlp. *ACM Computing Surveys*, 55(14s):1–39.

Alex Graves, Greg Wayne, and Ivo Danihelka (2014). Neural turing machines. *CoRR*, abs/1410.5401.

Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman (2009). *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, New York, NY.

Simon Haykin (1994). *Neural networks: a comprehensive foundation*. Prentice hall PTR.

Haiyun He, Yepeng Liu, Ziqiao Wang, Yongyi Mao, and Yuheng Bu (2024a). Theoretically grounded framework for llm watermarking: A distribution-adaptive approach. *CoRR*, abs/2410.02890.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick (2022). Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16000–16009.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen (2020). Deberta: Decoding-enhanced bert with disentangled attention. *CoRR*, abs/2006.03654.

Xinlei He, Xinyue Shen, Zeyuan Chen, Michael Backes, and Yang Zhang (2024b). Mgtbench: Benchmarking machine-generated text detection. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 2251–2265.

Yongxin He, Shan Zhang, Yixuan Cao, Lei Ma, and Ping Luo (2025). DETree: DEtecting human-AI collaborative texts via tree-structured hierarchical representation learning. In *Advances in Neural Information Processing Systems 38 (NeurIPS 2025)*.

Sepp Hochreiter and Jürgen Schmidhuber (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Kurt Hornik (1991). Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly (2019). Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR.

Jeremy Howard and Sebastian Ruder (2018). Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. (2022). Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3.

Jie Hu, Li Shen, and Gang Sun (2018). Squeeze-and-excitation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7132–7141.

Xiaomeng Hu, Pin-Yu Chen, and Tsung-Yi Ho (2023). Radar: Robust ai-text detection via adversarial learning. In *Advances in Neural Information Processing Systems*, volume 36, pages 15077–15095. Curran Associates, Inc.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer (2018). Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1875–1885.

Robin Jia, Aditi Raghunathan, Kerem Göksel, and Percy Liang (2019). Certified robustness to adversarial word substitutions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4129–4142, Hong Kong, China. Association for Computational Linguistics.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux,

Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed (2023). Mistral 7b. *CoRR*, abs/2310.06825.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits (2020). Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8018–8025.

Ian T. Jolliffe (2002). *Principal Component Analysis*, 2 edition. Springer Series in Statistics. Springer, New York, NY.

Daniel Jurafsky and James H. Martin (2026). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*, 3rd edition. Stanford University. Online manuscript released January 6, 2026.

Diederik P. Kingma and Jimmy Ba (2015). Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations (ICLR)*.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein (2023). A watermark for large language models. In *International Conference on Machine Learning*, pages 17061–17084. PMLR.

Kasper Thomas Gartside Knudsen and Christian Hardmeier (2025). Hybrid feature-embedding models for robust AI text detection. In *Proceedings of the 21st Conference on Natural Language Processing (KONVENS 2025): Long and Short Papers*, pages 318–325, Hannover, Germany. HsH Applied Academics.

Ron Kohavi (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI)*, volume 2, pages 1137–1143.

Ryuto Koike, Masahiro Kaneko, and Naoaki Okazaki (2024). OUTFOX: LLM-generated essay detection through in-context learning with adversarially generated examples. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 21258–21266.

Oliver Kramer (2016). Scikit-learn. In *Machine learning for evolution strategies*, pages 45–53. Springer, Cham.

Kalpesh Krishna, Yixiao Song, Marzena Karpinska, John Wieting, and Mohit Iyyer (2023). Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense. *Advances in Neural Information Processing Systems*, 36:27469–27500.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton (2015). Deep learning. *Nature*, 521(7553):436–444.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer (2020). Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 7871–7880.

Yafu Li, Qintong Li, Leyang Cui, Wei Bi, Zhilin Wang, Longyue Wang, Linyi Yang, Shuming Shi, and Yue Zhang (2024). MAGE: Machine-generated text detection in the wild. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 36–53, Bangkok, Thailand. Association for Computational Linguistics.

Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu (2022). A survey of transformers. *AI Open*, 3:111–132.

Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio (2017). A structured self-attentive sentence embedding. In *5th International Conference on Learning Representations (ICLR)*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov (2019a). Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov (2019b). Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Ilya Loshchilov and Frank Hutter (2019). Decoupled weight decay regularization. In *7th International Conference on Learning Representations (ICLR)*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning (2015). Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1412–1421.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu (2018). Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations (ICLR)*.

Marcus A Maloof (2006). Some basic concept of machine learning and data mining. In *Machine learning and data mining for computer security: Methods and applications*, pages 23–43. Springer, London.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean (2013a). Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean (2013b). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26.

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D Manning, and Chelsea Finn (2023). Detectgpt: Zero-shot machine-generated text detection using probability curvature. In *International conference on machine learning*, pages 24950–24962. PMLR.

Tom M Mitchell (1997). Does machine learning really work? *AI magazine*, 18(3):11–11.

Takeru Miyato, Andrew M. Dai, and Ian Goodfellow (2017). Adversarial training methods for semi-supervised text classification. In *International Conference on Learning Representations (ICLR)*.

John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi (2020). Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp. In *Proceedings of the 2020 conference on empirical methods in natural language processing: System demonstrations*, pages 119–126.

Kevin P Murphy (2012). *Machine learning: a probabilistic perspective*. MIT press.

Arsha Nagrani, Shan Yang, Anurag Arnab, Aren Jansen, Cordelia Schmid, and Chen Sun (2021). Attention bottlenecks for multimodal fusion. *Advances in Neural Information Processing Systems*, 34:14200–14213.

Joakim Nivre (2008). Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553.

Sinno Jialin Pan and Qiang Yang (2009). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

Jeffrey Pennington, Richard Socher, and Christopher D Manning (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

David MW Powers (2020). Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *CoRR*, abs/2010.16061.

Danish Pruthi, Bhuwan Dhingra, and Zachary C. Lipton (2019). Combating adversarial misspellings with robust word recognition. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5582–5591, Florence, Italy. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. (2018). Improving language understanding by generative pre-training. *mikecaptain.com*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. (2019a). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. (2019b). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Nils Reimers and Iryna Gurevych (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 3982–3992.

Leonard Rosenthol (2022). Content Credentials: C2PA Technical Specification. Technical report, C2PA. C2PA Technical Specification, Version 1.x.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams (1986). Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.

Vinu Sankar Sadasivan, Aounon Kumar, Sriram Balasubramanian, Wenxiao Wang, and Soheil Feizi (2023). Can AI-generated text be reliably detected? *CoRR*, abs/2303.11156.

V Sanh (2019). Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. In *Proceedings of Thirty-third Conference on Neural Information Processing Systems (NIPS2019)*.

Kristina Schaaff, Tim Schlippe, and Lorenz Mindner (2023). Classification of human- and AI-generated texts for English, French, German, and Spanish. In *Proceedings of the 6th International Conference on Natural Language and Speech Processing (ICNLSP 2023)*, pages 1–10, Online. Association for Computational Linguistics.

Konstantinos Sechidis, Grigorios Tsoumakas, and Ioannis Vlahavas (2011). On the stratification of multi-label data. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, pages 145–158.

Rico Sennrich, Barry Haddow, and Alexandra Birch (2016). Neural machine translation of rare words with subword units. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 1715–1725.

Shai Shalev-Shwartz and Shai Ben-David (2014). *Understanding machine learning: From theory to algorithms*. Cambridge university press.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani (2018). Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468.

Noam Shazeer (2020). GLU variants improve transformer. *CoRR*, abs/2002.05202.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean (2017). Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *5th International Conference on Learning Representations (ICLR)*.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola (2017). Style transfer from non-parallel text by cross-alignment. *Advances in neural information processing systems*, 30.

Marina Sokolova and Guy Lapalme (2009). A systematic analysis of performance measures for classification tasks. *Information processing & management*, 45(4):427–437.

Efstathios Stamatatos (2009). A survey of modern authorship attribution methods. *Journal of the American Society for information Science and Technology*, 60(3):538–556.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu (2024). Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler (2022). Efficient transformers: A survey. *ACM Computing Surveys*, 55(6):1–28.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. (2023). LLaMA 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Brian Tufts, Xuandong Zhao, and Lei Li (2025). A practical examination of ai-generated text detectors for large language models. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 4824–4841.

Adaku Uchendu, Zeyu Ma, Thai Le, Rui Zhang, and Dongwon Lee (2021). TURINGBENCH: A benchmark environment for Turing test in the age of neural text generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2001–2016, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Yuxia Wang, Jonibek Mansurov, Petar Ivanov, Jinyan Su, Artem Shelmanov, Akim Tsvigun, Osama Mohammed Afzal, Tarek Mahmoud, Giovanni Puccetti, Thomas Arnold, Alham Aji, Nizar Habash, Iryna Gurevych, and Preslav Nakov (2024a). M4GT-bench: Evaluation benchmark for black-box machine-generated text detection. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3964–3992, Bangkok, Thailand. Association for Computational Linguistics.

Yuxia Wang, Jonibek Mansurov, Petar Ivanov, Jinyan Su, Artem Shelmanov, Akim Tsvigun, Chenxi Whitehouse, Osama Mohammed Afzal, Tarek Mahmoud, Toru Sasaki, Thomas Arnold, Alham Fikri Aji, Nizar Habash, Iryna Gurevych, and Preslav Nakov (2024b). M4: Multi-generator, multi-domain, and multi-lingual black-box machine-generated text detection. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1369–1407, St. Julian's, Malta. Association for Computational Linguistics.

David H. Wolpert (1992). Stacked generalization. *Neural Networks*, 5(2):241–259.

Junchao Wu, Shu Yang, Runzhe Zhan, Yulin Yuan, Lidia Sam Chao, and Derek Fai Wong (2025). A survey on llm-generated text detection: Necessity, methods, and future directions. *Computational Linguistics*, 51(1):275–338.

Annepaka Yadagiri, Lavanya Shree, Suraiya Parween, Anushka Raj, Shreya Maurya, and Partha Pakray (2024). Detecting AI-generated text with pre-trained models using linguistic features. In *Proceedings of the 21st International Conference on Natural Language Processing (ICON)*, pages 188–196, AU-KBC Research Centre, Chennai, India. NLP Association of India (NLPAI).

Tong Zhou, Xuandong Zhao, Xiaolin Xu, and Shaolei Ren (2024). Bileve: Securing text provenance in large language models against spoofing with bi-level signature. *Advances in Neural Information Processing Systems*, 37:56054–56075.

Yichao Zhou, Jyun-Yu Jiang, Kai-Wei Chang, and Wei Wang (2019). Learning to discriminate perturbations for blocking adversarial attacks in text classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4904–4913, Hong Kong, China. Association for Computational Linguistics.