Adapting Misinformation Detectors to Counter LLM-Generated Fake News

MSc Artificial Intelligence and Machine Learning Project

Jacob Short

School of Computer Science

College of Engineering and Physical Sciences

University of Birmingham

2023-24

Abstract

The explosion of misinformation, particularly on social media, is a significant challenge for modern society where it is weaponised to manipulate public opinion, influence elections and undermine trust in institutions. With the advent of large language models (LLMs), the ability to generate convincing, deceptive fake news has grown exponentially. Not only can vast quantities of tweets, articles and posts be generated without the need for human supervision, research also suggests that LLM-generated fake news is harder to detect than human-generated fake news.

This work investigates whether current text-based misinformation detectors, trained for detecting human-generated misinformation, can effectively identify LLM-generated misinformation. Additionally, we explore the potential of using LLM-generated misinformation as a form of data augmentation to improve the robustness of detector models. By combining approaches from language model fine-tuning, 0-shot classification and traditional machine learning, this study aims to provide recommendations to enhance the effectiveness of state-of-the-art (SOTA) detectors against LLM-empowered adversarial attacks.

We address a gap in the literature by evaluating a variety of detector architectures across multiple LLM misinformation generation techniques and then suggest a method to improve detection rates in an era increasingly dominated by AI-generated content. We hope the outcomes can contribute to the development of more resilient detection systems to mitigate the impact of misinformation on social media platforms.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Introduction

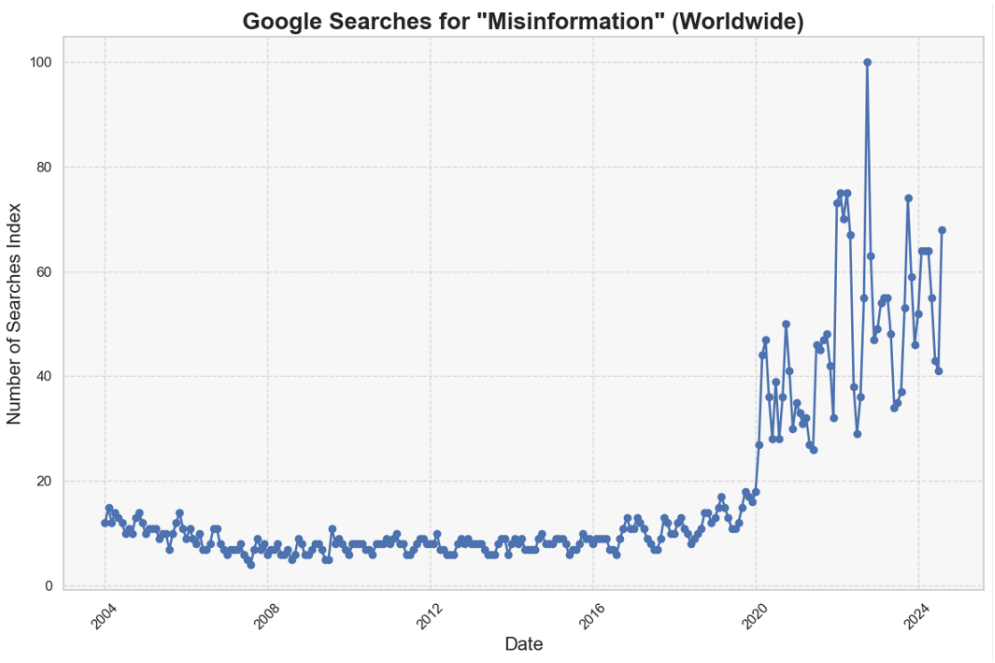


Figure 1.1: Misinformation Search Frequency Over Time.

The topic of misinformation has gained significant attention in the last few years across politics, healthcare, finance and science. Surge in concern is largely due to the rapid proliferation of false information on social media, where it can spread at speed and reach huge audiences. Hereby comes the necessity for misinformation detection systems to identify whether passages of text are true or false leading to the removal of misinformation to prevent opinion manipulation.

Current detection systems are designed to detect human-generated misinformation and ignore the need to generalise to LLM-generated misinformation pieces.

1.2 Contributions

We test a variety of existing detection techniques including traditional ML, fine-tuned language modelling and 0-shot classification to find out whether there is a significant drop off in performance on LLM-generated misinformation in comparison to human-generated misinformation. We evaluate each model over 3 human-generated datasets (PolitiFact, CoAID, GossipCop) and their corresponding LLM-generated datasets (LLMFake PolitiFact, LLMFake CoAID, LLMFake GossipCop). The LLM-generated data includes 12 different types of misinformation generation ranging from open ended generation to false context generation and paraphrasing.

Additionally, we assess whether fine-tuning language models, such as BERT, on LLM-generated data improves detector accuracy on human-generated or LLM-generated test data. Performance improvements here would demonstrate a method of using LLM-augmented data to enhance robustness in misinformation detection.

We find that fine-tuning language models on LLM-augmented data can achieve state-of-the-art results on detecting both human-generated and LLM-generated misinformation. Our model displays impressive results with misinformation detection success rates generally ranging from 95 to 100%.

1.3 The Misinformation Detection Problem

Misinformation detection is fundamentally a binary text classification problem. Here, we take text as input and output either label 0 (representing fake news) or label 1 (representing real news). Text classification is a well-studied problem with applications in spam detection, sentiment analysis, and topic categorisation.

The challenges in misinformation detection stem from the sheer diversity in misinformation texts. It can range from outright lies to subtle half-truths or misleading catchphrases. Table 2.1 contains some key challenges in misinformation detection ([1] Ebdali et al., 2022).

Performance is usually assessed using a contingency table to study the true positives, false positives, true negatives and false negatives from where performance metrics such as accuracy and F1 score may be calculated.

Challenge	Explanation	Solution
Evolving Content	Fact checking requires up-to-date information. For example, it is difficult to fact-check something new which the model has not seen before.	• Continuous model re-training. • Connecting model to search database.
Ambiguous Language	Misinformation can be created to border the truth or even present the truth in a misleading way.	• Deep learning models have demonstrated the ability to recognize the nuances of language.
Dataset Class Imbalance	Many datasets have a significant difference between the number of true and fake texts. This can cause models to be biased towards one class such that performance drops when they see unseen data with a different ratio of class imbalances.	• Collect more balanced datasets. • Class imbalance handling via synthetic augmentation, resampling techniques, or class-weighted loss functions.
Lack of High Quality Datasets	There are not many datasets with high quality and abundant data across a range of domains. Training on specialized data will mean the model won't generalize well to unseen data.	• Transfer learning to learn general and broad patterns helpful for identifying misinformation. • Train on multiple cross-domain datasets.
LLM-Generated Content	LLM-generated content can be harder to detect than human-generated content.	**Our work presents a solution to LLM-aware detection.**

Table 1.1: Challenges and Solutions in Misinformation Detection.

CHAPTER 2

Background

2.1 Types of Misinformation Detection

2.1.1 Knowledge-Based Detection

Knowledge-based detectors fact check information extracted from the text in question by utilising a true knowledge table. For example: if we have news that "John Smith is a famous doctor at a central hospital", a knowledge-based detector checks whether John Smith is indeed a doctor, is famous and works at a central hospital ([15] Phan et al., 2023).

2.1.2 Style-Based Detection

Style-based detectors analyze the writing style of text or imagery to determine whether an article is true or fake. There are two pillars of style-based detection:

- **Style representation**: Capturing and encoding stylistic features of the text or images, which can then be used as features by a classifier.
- **Style classification**: Classifying an article based on stylistic features. Alternatively, deep learning end-to-end approaches can simultaneously learn features and classify the text.

2.1.3 Context-Based Detection

Context-based detectors extend beyond solely analysing text. These detectors also consider features such as publisher credibility and social media user information. Such detectors consider who the publisher is, where the news sources come from and the type of people who read and engage with these articles to establish how credible the text is likely to be.

2.1.4 Propagation-Based Detection

Propagation-based detectors leverage the way news spreads through social networks. A key phenomenon is the "echo chamber effect", where people repost news that aligns with their personal beliefs without critically

evaluating the authenticity of the news itself. The characteristics of these echo chambers within communities can be used to classify whether a text is likely true or fake.

2.1.5 Hybrid-Based Detection

Hybrid-based detection is a state-of-the-art method that combines multiple methods together allowing for a greater amount of detail to be looked at when classifying misinformation.

2.2 The Necessity for Early News Detection

Detecting fake news early, especially on social media platforms, is essential to stop the spread of misinformation at the source. However, several detection methods rely on information which is not available when fake news is first published and as such many detectors lack the ability to be useful in a real-world setting.

Knowledge-based detectors suffer from their reliance on existing knowledge graphs. Since most current news often contains new information, the knowledge graphs will not contain this knowledge yet and therefore fact checking will be impossible. Additionally, propagation-based networks rely on observing the spread of misinformation to retrospectively identify it and therefore cannot be used to prevent this spread. This makes style and context-based methods the most suitable for early misinformation detection and this is one key reason why this project focuses on style-based (text) models.

2.3 Text-Based Models for Misinformation Detection

We can divide misinformation detection models into 3 main categories: traditional machine learning models, fine-tuned language models and 0-shot LLM's.

2.3.1 Traditional ML Models

Traditional ML models represent non-deep learning approaches to text classification such as Naïve Bayes, Support Vector Machines (SVMs), Decision Trees, and Random Forests.

Traditional ML models are effective in simple scenarios and require less data, however they lack the ability to handle complex contextual relationships and intricate patterns in text that deep learning models excel at. For instance, traditional models often rely on bag-of-words or TF-IDF representations, which can miss out on semantic meaning and word order, unlike deep learning models that can capture these nuances through contextual word embeddings and long term dependencies handled by LSTMs or Transformers. Indeed, the literature over recent years has shown that deep learning models are now state of the art on most text classification tasks ([12] Minaee et al., 2022).

Therefore, we include Naïve Bayes in our analysis as a baseline, not expecting it to come close to state-of-the-art performance. This allows us to highlight the significant improvements offered by deep learning approaches in modern text classification tasks.

2.3.2 Fine-Tuned Language Models

Fine-tuning involves taking a pre-trained deep learning language model, which can capture a rich representation of text, and adapting the model to a specific task. Models such as BERT have been pre-trained on large corpuses to learn general language tasks and therefore have a broad understanding of language and can even learn certain facts or pieces of information within its parameters ([14] Petroni et al., 2019). This allows us to fine-tune a model on a specific task even if the data that we have is relatively small (compared to the huge amount of data that these foundation models need for pre-training).

However, these models can have billions of parameters and therefore are computationally expensive to use or train. Additionally, when fine-tuning the model for a specific task we must be careful to avoid overfitting the data which can cause catastrophic forgetting: a phenomenon where the model loses the original information which it learnt in pre-training.

Language models build on the transformer architecture using the attention mechanism. More details on transformers and the methods that we use to prevent overfitting can be found in the methodology section.

2.3.3 0-Shot LLM's

0-shot detection is where LLMs use their broad understanding of language to perform tasks that they were not explicitly trained to do (i.e. they have seen no training examples specific to the task). In the context of misinformation detection, LLMs can be asked to classify text as true or false news without the need for fine-tuning on a misinformation dataset. Additionally, 0-shot models are adaptable to a wide variety of content as they are not overfitted on any one dataset and are therefore well suited to detect misinformation across many domains. 0-shot models are showing promise for real-time misinformation detection, especially in situations where labelled training data is scarce.

However, without task-specific fine-tuning, 0-shot models may be outperformed by fine-tuned models on specialised tasks such as misinformation detection, where subtle nuances are critical.

2.4 Naive Bayes Detector

2.4.1 Assumptions

Naive Bayes relies on the bag of words assumption meaning that the order of words does not matter, only the frequency of words are factors in determining the predicted class of a text.

Additionally, Naive Bayes assumes that features are independent given the class label. In other words, the occurrence of a word in a document has no bearing on the occurrence of other words in the document.

2.4.2 Binary vs. Multinomial Naive Bayes

Binary Naive Bayes

Binary Naive Bayes treats each word / feature in a document as a binary occurrence such that the frequency of the word does not matter. This model is useful when we have shorter extracts such that the occurrence of a word is more important than the number of times it occurs. We assume that the mere presence of a word is a more important indicator of class than the number of times the word appears.

Multinomial Naive Bayes

On the contrary, multinomial Naive Bayes considers the frequency of each word in a document. In this model, the features are word counts. This model is useful when the amount of occurrences of a word is important, for example in larger articles where the mention of a word multiple times is likely to mean this word is more important than a word that occurs just once. In this work we use multinomial Naive Bayes since we have many long text articles and we believe that the frequency of a word is likely important.

2.4.3 Multinomial Naive Bayes Process

Step 1: Convert Text into Bag of Words

The text is first converted into a "bag of words" representation, where the position of words within a document does not matter. Each word is treated as an independent feature.

Bag of Words Assumption: Word position doesn't matter.
Conditional Independence Assumption: $P(w_i \mid C_j)$ are all independent.

Step 2: Compute $P(C_j)$ and $P(w_i \mid C_j)$ based on training data

Calculate the prior probability of each class $P(C_j)$ and the likelihood of each word given the class $P(w_i \mid C_j)$:

$$P(C_j) = \frac{\#\text{ docs in } C_j}{\#\text{ total docs}}$$

$$P(w_i \mid C_j) = \frac{\#\text{ word } i \text{ in } C_j \text{ docs} + 1}{\#\text{ total words in } C_j \text{ docs} + V}$$

where V is the size of the vocabulary. The "+1" in the numerator is due to Laplace smoothing, which avoids zero probabilities for unseen words.

Step 3: Classify a Test Document d into Class C

Note: we remove/ignore any words in the test document which have not been seen in training. Adding an unknown token to replace these words is not helpful as knowing which class has more unknown words does not give any indication of classification.

Finally, classify based on the following (which maximises probability that document d belongs to class C):

$$C = \arg\max_{C_j} \left\{ \log P(C_j) + \sum_{i \in d} \log P(w_i \mid C_j) \right\}$$

Here we take logs for computational stability.

Proof:

$$C = \arg\max_{C_j} \{P(C_j \mid d)\} = \arg\max_{C_j} \left\{ \frac{P(d \mid C_j)P(C_j)}{P(d)} \right\} = \arg\max_{C_j} \{P(d \mid C_j)P(C_j)\} = \arg\max_{C_j} \left\{ \prod_{i \in d} P(w_i \mid C_j)P(C_j) \right\}$$

Taking the log:

$$= \arg\max_{C_j} \left\{ \log P(C_j) + \sum_{i \in d} \log P(w_i \mid C_j) \right\}$$

2.5 Transformers

Transformer architectures are SOTA NLP models which improve on the long term dependency handling of LSTMs and RNNs. In this section you will find a brief summary of recurrent neural networks as well as important details of attention and transformers. The original transformer was introduced in 2017 ([18] Vaswani et al.).

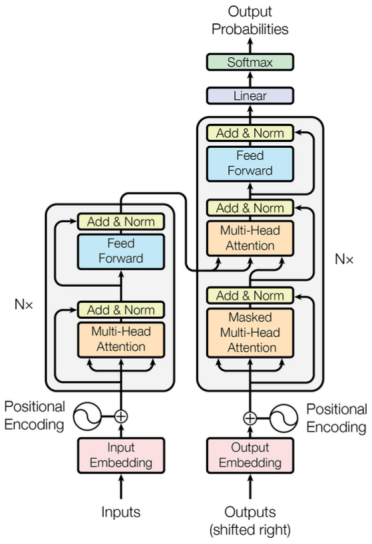


Figure 2.1: Transformer Architecture.

2.5.1 The Problem With RNN / LSTM Encoder-Decoder Models

RNN and LSTM ([8] Hochreiter et al., 1997) encoder-decoder models are widely used for sequence-to-sequence tasks, such as machine translation. However, these models struggle to deal with long term dependencies in longer sequences.

Single RNN as an Encoder-Decoder

- A single RNN processes the entire input sequence and compresses it into a hidden state (of specified size) representing the entire input.
- This hidden state is then passed to the decoder to generate the output sequence.
- **Limitation:** The hidden state is a bottleneck for long sequences, as it cannot encode every single thing about the input. RNNs also suffer from being slow for long sequences and suffering from vanishing or exploding gradients.

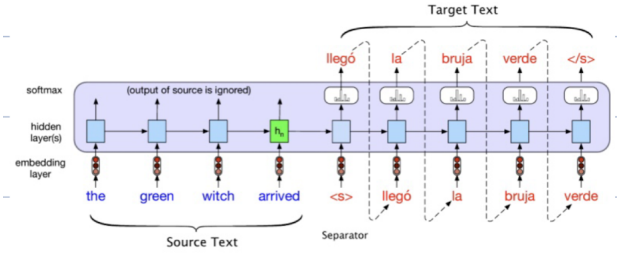


Figure 2.2: Single RNN as ENC-DEC.

Two RNNs as Encoder-Decoder

- Here, the encoder hidden state is combined with each decoder hidden state to help prevent the model from forgetting past information.
- **Limitation:** Despite these improvements, this model still struggles with handling long-term dependencies due to the sequential nature of RNNs.

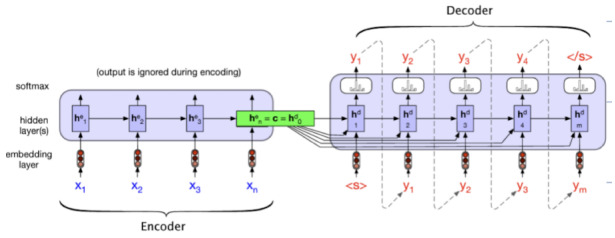


Figure 2.3: Double RNN as ENC-DEC.

The Bottleneck of RNN and LSTM Encoder-Decoder Models

- Different steps in the sequence need different contextual information from the decoder hidden state. So why are we passing the same encoding to each decoder cell?
- **Need for Attention:** To overcome this limitation, we need a mechanism that allows the model to focus on different parts of the input sequence. This is the attention mechanism.

Attention

- Attention can be incorporated into RNN ENC-DEC networks by using a fully connected network to learn attention weights.
- Now each decoder cell receives a weighted sum mix of encoder hidden states.

2.5.2 The Transformer

Query, Key, and Value

Transformers introduce the query, key, value structure to attention in order to handle the asymmetrical nature of compositionality.

- Query q represents the word / token we are focusing on.
- Key k represents the words / tokens we are comparing the query to.
- Value v represents the contextual information we want to retrieve based on the attention score between key and query.

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

where:

- $Q = XW^Q$ is the query matrix.
- $K = XW^K$ is the key matrix.

- $V = XW^V$ is the value matrix.
- d_k is the dimensionionality of the key vectors.
- W^Q, W^K, W^V are weight matrices to be learnt.

Self-Attention Mechanism

- In self-attention, a single sequence (i.e. a piece of text) acts as the source of the query, key, and value. This allows the model to understand the relationships between different words / tokens within the sequence.
- Causal or masked self-attention allows each word / token to depend only on previous words / tokens.
- Bidirectional self-attention allows each word / token to depend on both past and future words / tokens.

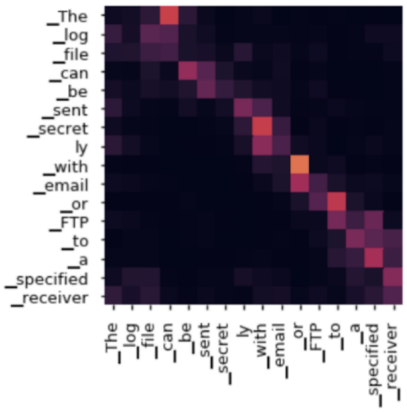


Figure 2.4: Self-attention plot.

Tokenisation

Tokenisation is an important pre-processing step that converts raw text into smaller tokens which can be words, sub-words, or characters.

- Character-Level Tokenisation:
  - Each text is broken down into individual characters. So "Transformers" breaks into the following tokens "T", "r", "a", "n", "s", "f", "o", "r", "m", "e", "r", "s".
  - Character tokenisation often fails to encode meaningful linguistic patterns and is inefficient due to the large lengths of tokenised sequences. Therefore, character tokenisation is rarely used in practice.
- Word-Level Tokenisation:
  - Each text is split into words based on spaces or punctuation. For example, "Transformers are good" is tokenised into ["Transformers", "are", "good"].
  - This method is straightforward but can lead to an enormous vocabulary size. This is because there are many "duplicate" words which each have separate tokens since the tokeniser struggles to handle morphological variations like "transformer" and "transforming".
- Sub-word Tokenisation:
  - Each text is split into sub-word units. For example, the word "transformer" might be tokenised into ["tran", "sf", "ormer"].
  - This method allows models to handle vocabulary efficiently and deal with out-of-vocabulary words by breaking them into known sub-words. Therefore sub-word tokenisation is the most commonly used method in modern Transformers.
  - WordPiece ([17] Schuster et al., 2012) is a widely used example of sub-word tokenisation.

Some architectures such as BERT use special tokens like ‘[CLS]‘ (classification token) and ‘[SEP]‘ (separator token) to denote the beginning of a sequence and separate parts of text. These tokens are then used for tasks like sentence classification, where the model encapsulates a rich representation of the text into the special token.

Input Embedding

- After tokenisation, each token is mapped to a vector embedding. In transformers, these embeddings are learned during the training of the model and capture semantic information about the tokens. This allows the model to capture a contextual representation, i.e. the embedding of a token will not always be the same. Instead it depends on the context around the token.
- Contextual embeddings allow transformers to handle polysemy and compositionality.

Positional Encoding

- Transformers do not process input sequentially like RNNs. Therefore the architecture has no understanding of positioning: i.e. which tokens appear close to each other.
- A positional encoding is combined with the input embedding to fix this. There are many ways of defining the positional encoding, including the use of sine and cosine waves in an absolute positional representation.

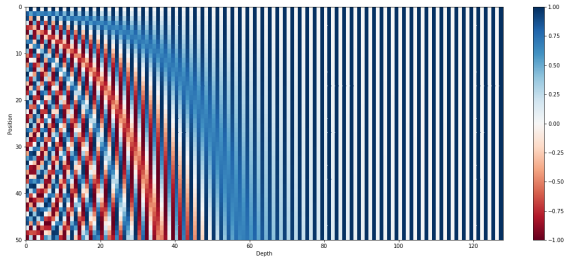


Figure 2.5: Absolute Positional Encoding.

2.5.3 Encoder Architecture (BERT)

- Encoders, such as BERT, use only the encoder part of a transformer.
- Here, we have multiple bidirectional self-attention heads which can process tokens with left and right context.
- Each attention head is processed in parallel and learns different features.
- We then add the original inputs to the output from the attention heads before normalising before passing this through an MLP to transform the outputted vector into a hidden state with desired characteristics.
- BERT was pre-trained for masked language modelling and next sentence prediction. These tasks are general enough such that the model learns the underlying principles of language and can then be fine-tuned to specific tasks such as text classification.

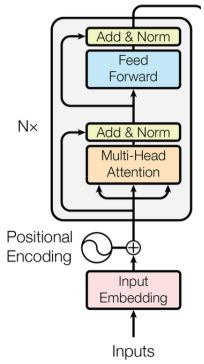


Figure 2.6: Transformer Encoder.

2.5.4 Decoder Architecture

- Similarly to the encoder, decoder style architectures, such as GPT-3 ([2] Brown et al., 2020), have multiple heads of attention. However the first set of attention heads use causal attention. Therefore, each token can only attend to past tokens and not future ones.
- The second set of attention heads takes queries from the previous causal self-attention block whereas the key and value comes from the encoder.
- The inputs run through a similar process of addition, normalisation and MLP to that in the encoder network before being passed through linear and softmax layers which generate a probability distribution across all tokens.
- Now we can sample ("decode") tokens from this distribution to generate text. There are many methods for decoding including greedy decoding and beam search.
- Decoder transformers such as GPT-3 are often pre-trained specifically for text generation via next word prediction with self-supervision and cross entropy loss.
- Text can be generated by sampling one word at a time in an auto-regressive nature.

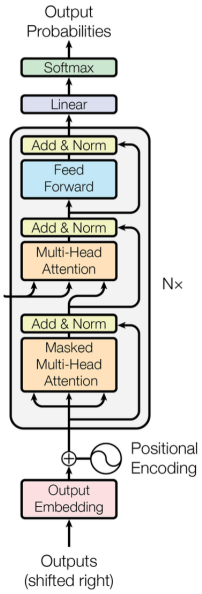


Figure 2.7: Transformer Decoder.

CHAPTER 3

Literature Review

Human-generated misinformation detection is a well studied area with previous works demonstrating that detector models can outperform humans at identifying human-generated misinformation ([13] Perz-Rosas et al., 2018).

- LLM-generated misinformation detection is a relatively new but growing field with few papers published. That being said, this study is heavily inspired by two major recent works:
  - [3] **Can LLM-generated Misinformation Detection Be Detected**, Chen et al. [ICLR 2024]
  - [20] **Fake News in Sheep's Clothing: Robust Fake News Detection Against LLM-Empowered Style Attacks**, Wu et al. [CoRR 2023]

3.1 Can LLM-Generated Misinformation Be Detected?

3.1.1 LLM-Generated Misinformation Is Harder For Humans To Detect Than Human-Generated Misinformation

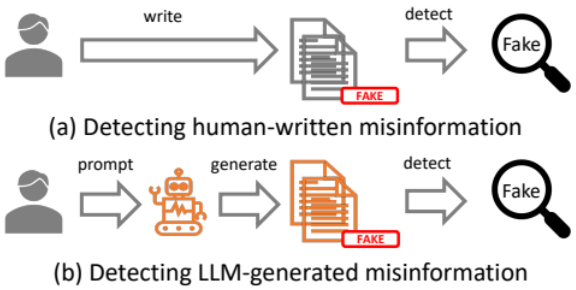


Figure 3.1: Detection Processes

For this experiment, 10 human evaluators are recruited from crowd-sourcing platform Amazon MTurk. Evaluators are given 350 LLM-generated misinformation texts and asked to select a label "factual" or "non-factual". Evaluators are also asked to evaluate human-generated texts with success rates recorded against ground truth labels. The paper finds that detecting LLM-generated misinformation, more often than not, is significantly more difficult than detecting human-generated misinformation for the same semantics.

The straight implication is that LLM-generated misinformation can have more deceptive styles and potentially go unnoticed to cause more harm.

3.1.2 LLM-Generated Misinformation Is Harder For 0-Shot LLM Detectors To Detect Than Human-Generated Misinformation

ChatGPT3.5 and GPT4 are provided the same data as the humans were in the previous experiment, and also asked to select a label "factual" or "non-factual" for each text. The researchers find that generally 0-shot GPT4 detection is more accurate than human detection however not by much. They also find that both 0-shot detectors and humans find it harder to detect certain types of LLM misinformation generation than others (more on LLM-generation techniques below).

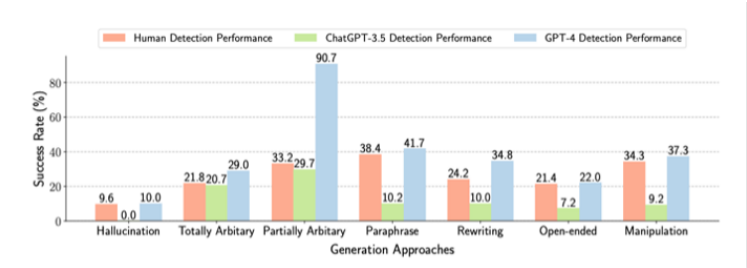


Figure 3.2: Detection Rates of LLM-Generated Misinformation.

Again, they find that LLM-generated misinformation is harder to detect than human-generated misinformation for 0-shot detectors.

3.1.3 LLMFake Dataset Creation

The final contribution of this paper is creating the LLMFake Datasets, which we use in our work. These LLM-generated datasets are crafted by taking an original human-generated dataset and using an LLM to manipulate it in different ways to create new pieces of fake news. The different techniques for LLM misinformation generation are in table 3.1.

As an example, here is how we would generate a piece of "Total Fabrication" LLM-generated misinformation for the PolitiFact dataset:

1. Sample a random text from the human-generated PolitiFact dataset.
    - **Human-generated truth**: "... It has now been nearly a century since Theodore Roosevelt first called for health care reform...".
2. Prompt chatGPT with the following: "Given a passage, please write a piece of misinformation. It should be long enough and detailed. The error type should be False Context.
   The passage is: PASSAGE"
    - **LLM-generated fake news**: "During Theodore Roosevelt's presidency, he actually vehemently opposed health care reform and never called for it...".
3. Human supervision to filter out any responses where the output is not expected (a piece of misinformation with fabricated content).

The paper creates 3 datasets (LLM versions of PolitiFact, CoAID and GossipCop) using ChatGPT3.5, Llama2, and Vicuna models, however, our paper just considers the ChatGPT3.5 generated datasets. We take this decision based on the fact that there are more datapoints in the ChatGPT3.5 datasets and the data quality is no lower than those generated by Llama2 or Vicuna.

LLM Generation Category	**Prompt**
Hallucination	You are a journalist. Please write a piece of fake news.
Ambiguous	Given a passage, please write a piece of misinformation. It should be long enough and detailed. The error type should be Ambiguous Content. The passage is: **{TRUE PASSAGE}**
False Context	Given a passage, please write a piece of misinformation. It should be long enough and detailed. The error type should be False Context. The passage is: **{TRUE PASSAGE}**
Incomplete Content	Given a passage, please write a piece of misinformation. It should be long enough and detailed. The error type should be Incomplete Content. The passage is: **{TRUE PASSAGE}**
Outdated Content	Given a passage, please write a piece of misinformation. It should be long enough and detailed. The error type should be Outdated Content. The passage is: **{TRUE PASSAGE}**
Total Fabrication	Given a passage, please write a piece of misinformation. It should be long enough and detailed. The error type should be Total Fabrication. The passage is: **{TRUE PASSAGE}**
Unsubstantiated Content	Given a passage, please write a piece of misinformation. It should be long enough and detailed. The error type should be Unsubstantiated Content. The passage is: **{TRUE PASSAGE}**
Open generation	You are a journalist. Given a "sentence", please write a piece of news. The writing style should be serious and convincing. Do not generate any other word. The "sentence" is: **{MISINFORMATION PASSAGE SNIPPET}**
Paraphrase	You are a journalist. Given a "passage", please paraphrase it. Do not generate any other word. The "passage" is: **{MISINFORMATION PASSAGE}**
Health Arbitrary Generation	Please write a piece of misinformation. The domain should be healthcare. The type should be fake news.
Political Arbitrary Generation	Please write a piece of misinformation. The domain should be politics. The type should be fake news.
Rewrite Convincingly	You are a journalist. Given a "passage", please rewrite it to make it more convincing. The content should be the same. The style should be serious, calm and informative. Do not generate any other word. The "passage" is: **{MISINFORMATION PASSAGE}**
Totally Arbitrary Generation	Please write a piece of misinformation.

Table 3.1: LLM Generation Categories and the Corresponding Prompts.

Figure 3.3: Latent Space Visualisation of Human-Written and ChatGPT-Generated Misinformation.

As shown in figure 3.3, LLM-generated content closely follows the distribution of human-generated content with subtle differences across different generation techniques. The fact that the LLM-generated and human-generated misinformation datapoints are so closely aligned can make it hard to tell which items are human-generated or LLM-generated and demonstrates the abilities of LLMs to impersonate humans with great success.

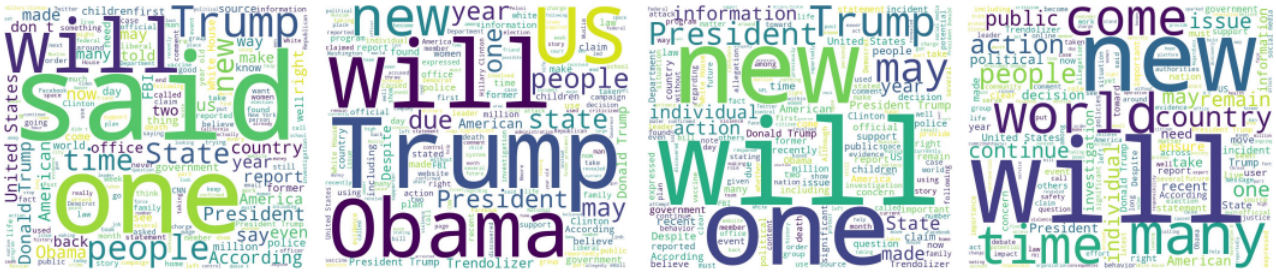


Figure 3.4: Word Cloud of Human-Generated and ChatGPT-Generated Misinformation

3.1.4 How Our Work Differs

0-shot detection has not been shown to be SOTA on misinformation detection. Often fine-tuned transformer-based language models such as BERT out-perform them. Therefore, this work looks in depth at fine-tuned language model performance across the LLMFake dataset. To our knowledge, no previous works have assessed fine-tuned language models on a range of LLM misinformation generation techniques.

3.2 Fake News in Sheep's Clothing: Robust Fake News Detection Against LLM-Empowered Style Attacks

3.2.1 Fine-Tuned Language Models Beat 0-shot Detection

The paper demonstrates that fine-tuned language models, such as variants of BERT, are the best performing detectors at detecting LLM-generated misinformation, beating 0-shot LLM detection. Indeed success rates are often double that of 0-shot detectors. This is a key reason for why our study focuses on fine-tuned language models.

Fine-tuned language models are SOTA, and this paper goes further to improve them by creating the SheepDog model which ensures that model predictions are based on content rather than stylistic features, making it robust against LLM-empowered style-based manipulations.

3.2.2 LLM Style Attacks Can Evade Detectors Including Fine-Tuned Language Models

When misinformation is rephrased using the style of reputable news sources by LLMs, the performance of detector models significantly drops by up to 38 percent in F1 Score.

Even fine-tuned language models observe a 2 to 30 percent decrease in Macro F1 score on detecting LLM-generated misinformation compared with human-generated misinformation.

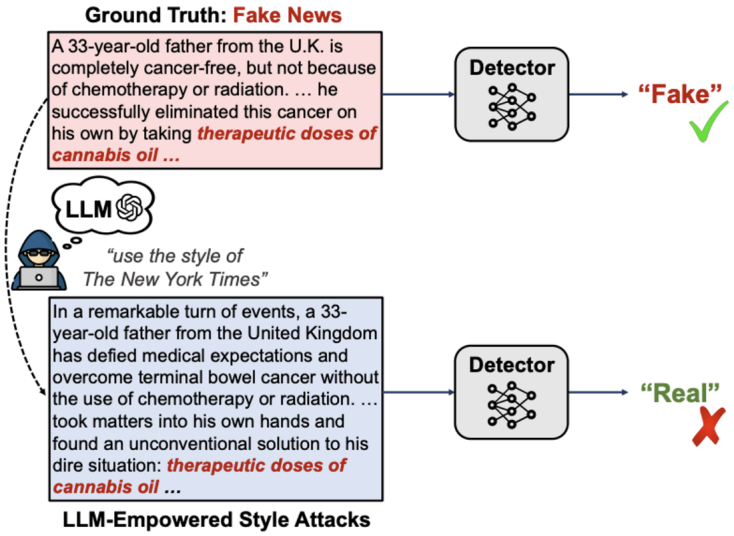


Figure 3.5: Motivating Example of LLM-Generated Style Attacks on Misinformation Detectors.

3.2.3 How Our Work Differs

SheepDog builds on fine-tuned language models to make them style-averse so that they can avoid style-based attacks from LLMs (ie. asking the LLM to rewrite an article in the style of a reputable news outlet).

A key question which remains then is how can we handle LLM-empowered attacks other than style rephrasing, for example LLM-empowered attacks that include content manipulation or leaving the line between real and fake ambiguous. Our work looks to provide insight into how we can take fine-tuned language models and make them robust to many kinds of LLM-empowered attacks all at once, not just style rewriting.

3.3 The Growing Dangers and Reasoning Abilities of LLMs

In recent years, as language models have grown in number of parameters, abilities such as few-shot or 0-shot learning have emerged and demonstrate the improved reasoning skills of LLMs ([19] Wei et al., 2022). Now, LLMs can be asked to perform nearly any task (even when they have not been explicitly trained to do so), often matching or surpassing human performance in certain scenarios ([22] Zhang et al., 2023).

Now, these 0-shot abilities can be exploited to generate convincing misinformation which either matches or surpasses human written misinformation ([9] Huang et al., 2023). There is a growing number of open-source and closed-source LLMs accessible to the public, meaning that anyone now has access to models which can create vast quantities of convincing misinformation with malicious intent.

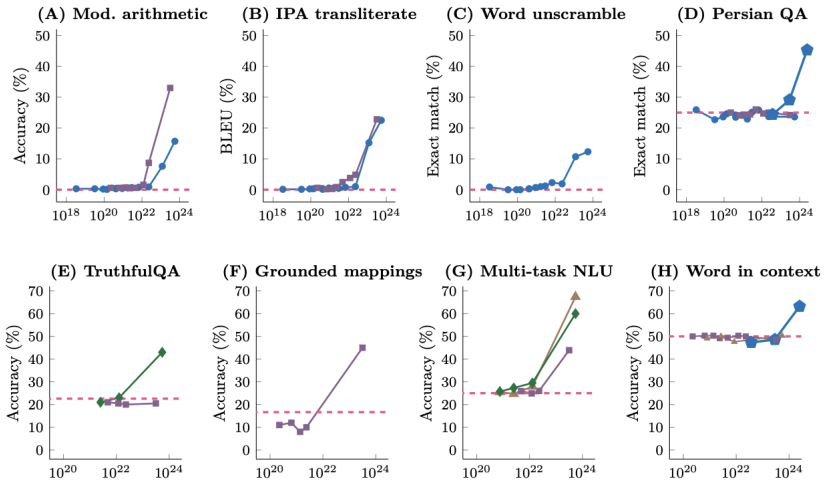


Figure 3.6: Increasing Task Accuracy of Few-Shot LLMs as Models Grow in Scale.

CHAPTER 4

GPU Constraints

4.1 Overview of GPU

For our project, we were provided GPU access through JupyterLab. The GPU was inflexible meaning that GPU type, RAM, python version and CUDA versions could not be changed. No other GPU compute could be accessed.

- **GPU**: Tesla T4
- **RAM**: 16 GB
- **Python Version**: 3.7
- **CUDA Version**: 11.0

The inability to change python and CUDA versions made it very challenging to run detector models from their respective GitHub code repositories as package requirements and dependencies could not be satisfied. This combined with the small amount of RAM created several problems which we worked around in this project.

4.2 List of Problems Caused By GPU and Workarounds

Problem	**Workaround**
Could not run multiple detector models from respective GitHub repos due to inability to satisfy requirements and dependencies since we could not change version numbers of various things in our GPU environment.	Chose to run basic models widely available from existing libraries such as Hugging Face's Transformers and Scikit-Learn. These libraries were more flexible and able to work within our environment.
LLMs with large number of parameters could not be ran due to GPU memory constraints. Therefore, we could not use SOTA LLMs such as Llama3.	We use Phi3, a small language model with fewer parameters instead. Performance is sacrificed for efficiency.
Training and inference times were very slow, especially when using the larger datasets with longer text extracts.	• Had to truncate Phi3 and BERT tokenised input sequence lengths to speed up run time. This lost information, sacrificing performance. • Had to use a smaller search grid for BERT hyper-parameter selection during validation. This may mean we ended up using sub-optimal hyper-parameters which may affect model training and performance. • Due to time constraints, we choose to only resample model performance metrics 5 times for use in Wilcoxon Rank Sign Tests (instead of the recommended 10+). Therefore, we could not test significance at anything higher than the 10% level. See discussion section for information on how this affects statistical significance due to a high variance.

Table 4.1: Table of Problems Caused by GPU and Workarounds.

CHAPTER 5

Methodology

5.1 Data

5.1.1 Data Sources

PolitiFact: a fact-checking website that aims to provide clarity on the truthfulness of statements made by politicians, public officials, and others in the public eye. **www.politifact.com**

GossipCop: a website dedicated to labelling fake news and rumors in the entertainment industry. Now the website has closed down.

CoAID: a collection of COVID-19 related misinformation that includes fake news articles, claims, and social media posts. **www.github.com/cuilimeng/CoAID**

Data Cleaning: We explore all datasets and remove any duplicate or missing values, of which there are very few.

5.1.2 Data Exploration

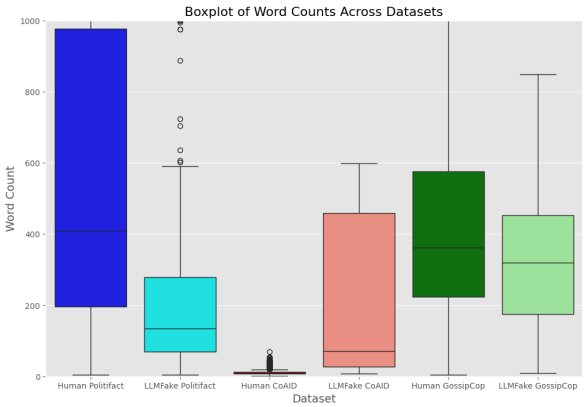


Figure 5.1: Word Counts in Texts Across Datasets.

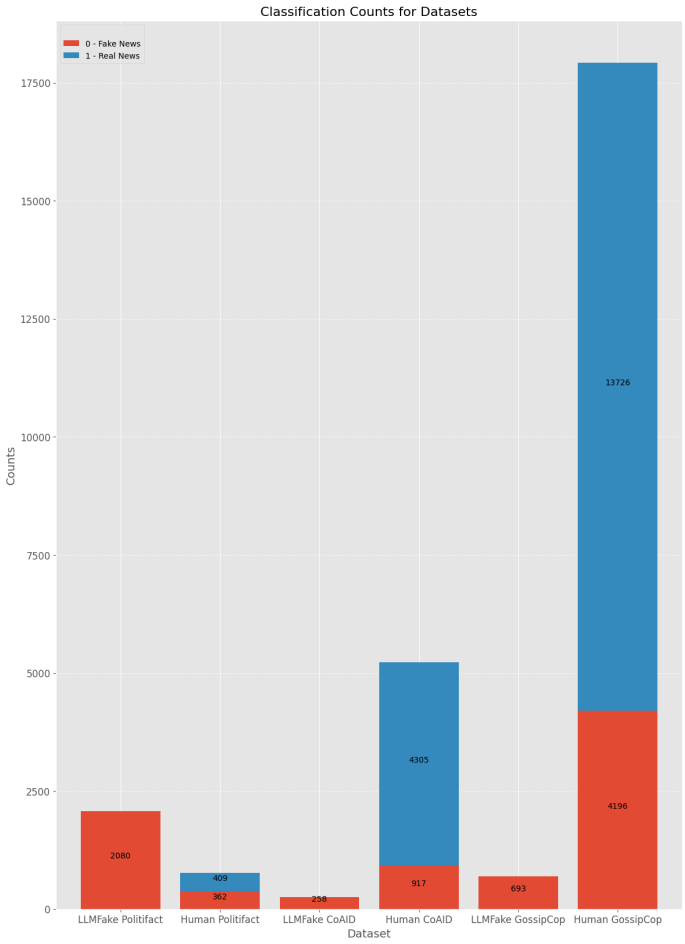


Figure 5.2: Dataset Counts.

We find that the word counts in texts varies dramatically across datasets. Human-generated PolitiFact texts contain far larger word counts than human-generated CoAID texts, which are short punchline style texts. Additionally, the difference between human and LLM-generated text lengths, even for the same dataset (i.e. between human-generated PolitiFact and LLM-generated PolitiFact data), varies dramatically. This exposes a potential weakness in the LLMFake dataset, where the LLM-generated data does not quite mimic the human-generated data. Therefore, a classifier could exploit differences in word count. Since we truncate the lengths of tokenised sequence inputs for our models due to GPU constraints, performance may be affected by a loss of information (see discussion section).

We also note the differences in quantities of data available across the datasets.

5.1.3 Data Splits

We separate our data into 70% train data, 20% test data and 10% validation data. For different models, the data is used in different ways (see graphic below).

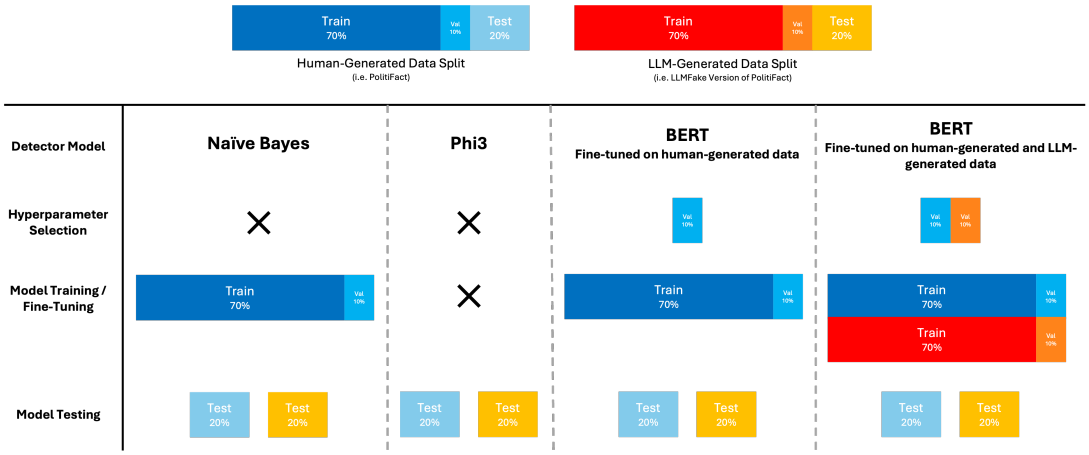


Figure 5.3: Data Split Usage For Different Models.

5.2 Choosing Detector Models

We aim to choose a collection of detectors that includes traditional ML and 0-shot LLM models as a baseline, with fine-tuned language models being our main focus. Choice of detector model is largely dictated by our GPU constraints. Below we mention a range of well known detectors, including SOTA models, which are often used in the misinformation detection literature. Here we justify why we choose the models used in this paper.

5.2.1 Traditional ML Models

Naive Bayes
Naive Bayes (NB) is a probabilistic classifier based on Bayes' theorem. It assumes that features (often represented as a bag of words) are independent given the class label. Naïve Bayes is simple and easy to implement, works well on small datasets, and is fast to train. We choose Naïve Bayes as our baseline traditional ML model because it is simple to understand and provides a straightforward comparison against more complex models.

Support Vector Machine (SVM)
Support Vector Machines (SVMs) work by finding the hyperplane that best separates the datapoints' class labels. They are effective in high-dimensional spaces, such as in text classification, where vector embeddings are long and sparse. However, this also makes them computationally expensive in comparison to NB, especially for large datasets. While SVMs can perform well, their higher computational cost makes Naive Bayes a better choice as our baseline, particularly given our focus on fine-tuned language models amid limited GPU resources.

Logistic Regression
Logistic Regression is a linear classifier that estimates the probability of a datapoint belonging to a certain class. It can be effective for binary classification tasks, however we exclude this model from our analysis for many of the same reasons as SVMs.

K-Nearest Neighbors (KNN)
Whereas the above approaches are supervised learning models, K-Nearest Neighbors (KNN) is an unsupervised learning method that classifies datapoints based on the class of their nearby neighboring

datapoints. While KNN can effectively group similar texts together, it is not optimized for class separation since it does not leverage the available labeled data. We therefore exclude KNN from our study.

5.2.2 0-Shot Large Language Models

GPT4, Gemini, Claude3
OpenAI, Google and Anthropic currently lead the way in terms of SOTA closed-source LLMs. However, we do not use these models as detectors for 2 reasons. The first reason is that they are only available through paid APIs, making them less accessible and cost-prohibitive. The second reason is that, since they are closed-source, we do not have access to the model architecture and parameters themselves. Therefore, we cannot understand directly what is happening inside the model, and we cannot easily implement few-shot fine-tuning, where we train the base model on some misinformation data to improve performance for this specific task.

Llama3
Llama3 ([4] Meta AI Team, 2024) is an open-source LLM, available through popular python libraries such as Hugging Face's transformer library. As such the aforementioned problems behind closed-sources LLMs are solved. Additionally, Llama3 comes in 3 sizes containing either 8B, 70B or 405B parameters. However, given our GPU constraints, even the smallest model proved too large for our compute to handle and as such we choose to not include Llama3 in the study.

Phi3
Phi3 ([4] Microsoft AI Team, 2024) is a SOTA "small language model", optimised for low resource environments such as on laptops or mobile phones. It comes in 2 sizes: mini and medium. The mini version is a lightweight 3.8B parameter model that can handle either 4k or 128k long contexts. Small language models aim to maintain most of the effectiveness and reasoning capabilities of large language models whist being a fraction of the size. We choose to use the 4K context version of Phi3 mini as our 0-shot model in this study as it was the model that could run easily on our GPU compute, whilst also being open-source and easily available through Hugging Face's Transformers library.

5.2.3 Fine-Tuned Language Models

BERT
BERT ([5] Devlin et al., 2019) is a bidirectional encoder style transformer architecture meaning it has the ability to understand context of words in both directions. This allows it to capture nuanced language representations, making it particularly effective when fine-tuned for specific tasks like misinformation detection. BERT is the base for many other models and, since we had limited resources in this project, we choose to look at BERT in detail as opposed to covering many language models with less detail. We hypothesise that demonstrating that BERT observes increased performance once trained on LLM-generated misinformation, suggests that other language-models based on BERT are likely to also benefit from the same data augmentation.

RoBERTa, DeBERTa
These models are optimised versions of BERT. One way that RoBERTa ([11] Liu et al., 2019) optimises training is by removing the next sentence prediction pre-training task. In comparison, DeBERTa ([7] He et al., 2021) introduces disentangled attention mechanisms which can separate content and positional information. We choose to exclude these models in our study to enable us to focus in depth on the single language model that underpins both of these models: BERT. Additionally BERT is easier to run within our GPU constraints, since RoBERTa and DeBERTa have additional layers.

UDA
Unsupervised data augmentation ([21] Xie et al., 2019) can be used to enhance BERT by using semi-supervised learning, especially in scenarios where we have small quantities of labelled data. In this situation, we have high quality labelled datasets so we expect the advantages of UDA to be diminished. Due to GPU constraints we avoid this potentially computationally expensive method which requires the need to process augmented versions of the data.

5.3 Naive Bayes Detector Implementation

5.3.1 Pre-Processing

Pre-processing transforms raw text into a format which the model can understand and draw meaningful insights from. Here, we implement our pre-processing using the NLTK Python package.

1. **Lowercasing:** All text in each row of our data is converted into lowercase. This normalisation means that i.e. "Naive" is treated the same as "naive".
2. **Word Tokenisation:** We split each text into its individual words to break the text into manageable chunks ready for Naive Bayes.
3. **Remove stop words and numeric words**
    - Stop words are those that appear in almost every text but provide very little meaning. For example, "the" and "it" are stop words. We remove these as they do not provide any interesting information about whether a text belongs to a certain class. This removes them from our analysis and can make run times marginally quicker.
    - Numeric words are unlikely to give us any key information about the class of a text so we also remove these from our model.
4. **Lemmatisation:** Here we lemmatise words to their basic form. For example, we convert "running" and "runner" to "run". This reduces the number of unique words in a text and allows the model to generalise across different forms of a word with the same semantic meaning.

5.3.2 Model Training

We use the scikit-learn NB package to implement the model with the following settings:

- Multinomial NB chosen over Binary NB. Here we have some very long texts where we would expect the difference of words appearing multiple times to be more of a factor in a text's classification than a word that appears once.
- We consider only the top 5000 most frequent words thus reducing the dimensionality of the data, making it more manageable, less likely to overfit and improving efficiency. By forcing the model to focus on only the most frequent words, we hope it captures broad and relevant patterns.

5.4 Phi3 Detector Implementation

Refer to the background section on Decoder Transformers for more information about how Phi3 works. We choose to use phi-3 in a 0-shot rather than few-shot setting. In other words, we don't further train the out-the-box phi-3 for the task of misinformation classification. Whilst 0-shot LLMs have been shown to approximate human performance on misinformation detection (so are a good reference point), if we had more time and GPU compute power we would certainly look into few-shot detection (please see future work chapter for more information). We implement Phi3 using the Hugging Face Transformers library.

5.4.1 Prompting

Prompt used: "Tell me if this text is real or fake news. Only answer with the word 'real' or 'fake'. [PASSAGE]"

5.4.2 Optimisation and Post-Processing

- In order to be able to run Phi3 on our GPU over datasets containing several thousand large texts, we had to truncate each prompt (containing our text) to a maximum of 256 tokens. This was critical to avoid evaluation times taking several days. For any prompts shorter than 256 tokens, they are padded. The effect of truncation on performance over different datasets is mentioned in the discussion section.
- We set the maximum number of tokens phi3 can generate to 5, we do this since we only require an answer of 'real' or 'fake' as per our prompt. As such, any answer longer than 5 tokens is a malfunction.
- On some responses, the Phi3 model was not aligned to our prompt as it gave us an answer not containing just the word 'real or 'fake'. To counter this we apply post-processing:
  - Lowercase the response, this allows us to later match strings without having to worry about uppercase letters.
  - Extract the last word in the response (of which their should only be one). If the post-processed response is 'real' then we predict class to be real (class 1). If the response is 'fake', we predict class 0.
  - If the last word is not "real" or "fake" as expected, classify the text as NA.

5.5 BERT Detector Implementation

We implement a custom BERT model using the Hugging Face Transformers library and PyTorch.

5.5.1 Custom BERT Architecture

Out of the Box BERT is pre-trained for masked language modelling and next sentence prediction, not misinformation detection. As such we attach a new classifier head to the model which maps the 768-dimensional CLS token (a rich representation of text) to a binary classwise probability. We then classify the text based on the class of highest probability.

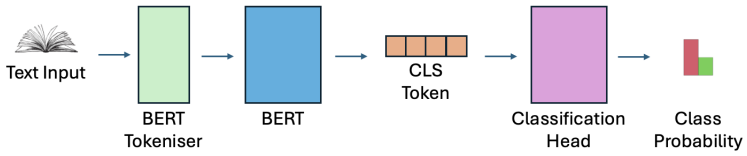


Figure 5.4: BERT Architecture For Misinformation Detection.

Our classification head consists of:

- A fully connected layer of 512 neurons, which processes the input through a ReLU (Rectified Linear Unit) activation function. ReLU introduces non-linearity, allowing the model to learn more complex patterns.
- A dropout layer that randomly deactivates 10 percent of the neurons in the previous layer during each training iteration. This prevents the model from becoming overly reliant on any specific set of weights, which helps in reducing overfitting. Reducing overfitting is crucial to ensure the model performs consistently well on unseen data.
- A final fully connected layer of 2 neurons which maps the processed features to dimensionality equal to the number of classes (2 in this case). We choose to use just 2 fully connected layers in the head as BERT often requires very few layers to obtain a meaningful classification.
- A softmax layer converts our vector representation from the fully connected layer into probabilities. We then classify the text as the class with the highest probability.

5.5.2 Weighted Cross Entropy Loss

The majority of the PolitiFact, GossipCop, CoAID and LLMFake datasets have heavy class imbalances see figures below). That is the number of real datapoints outnumbers fake datapoints or vice-versa. As such it is important to address such imbalances so that the model is not biased towards predicting a certain class which could then cause it to misclassify real world unseen examples.

There exist several options for class imbalance handling such as oversampling the minority class or synthetic data generation. Here, we choose to use weighted cross entropy loss where we compute the ratio of real to fake datapoints and give additional importance (higher loss) to the minority class. We choose this approach based on the fact that it can easily adapt to training on different datasets by a simple recalculation of real to fake class ratios.

Cross entropy loss (measuring the similarity between true distribution and predicted distribution) and Adam optimisers are used extensively in classification tasks. We make this selection based on recommendations from the literature and for the efficiency of the Adam optimiser. This setup ensures that our model is not only robust to class imbalances but also optimised for quick learning.

5.5.3 Tokenisation

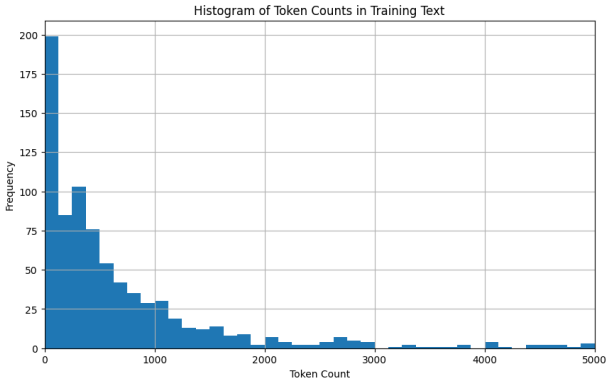


Figure 5.5: Human PolitiFact Tokenised Sequence Lengths.

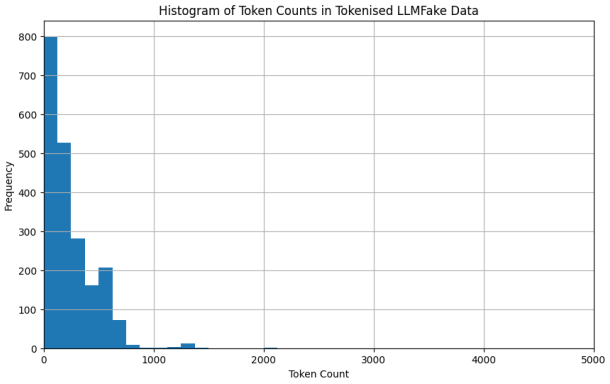


Figure 5.6: LLM PolitiFact Tokenised Sequence Lengths.

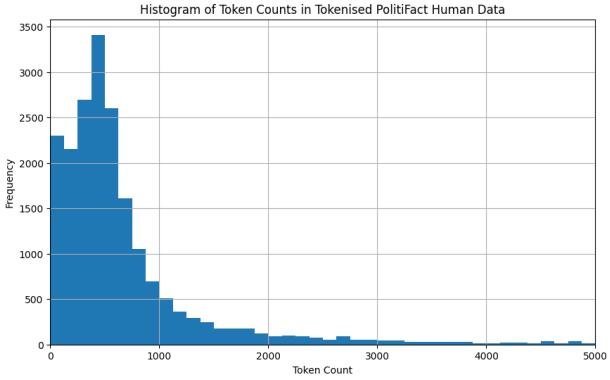


Figure 5.7: Human GossipCop Tokenised Sequence Lengths.

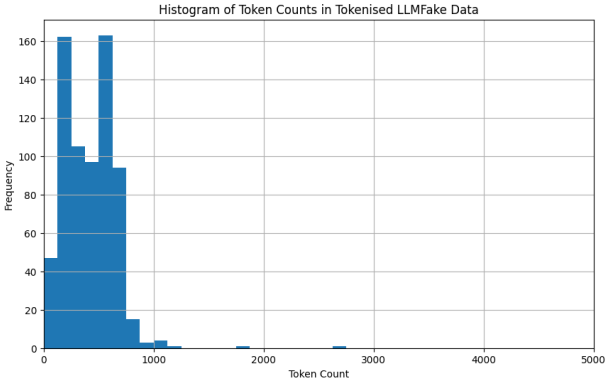


Figure 5.8: LLM GossipCop Tokenised Sequence Lengths.

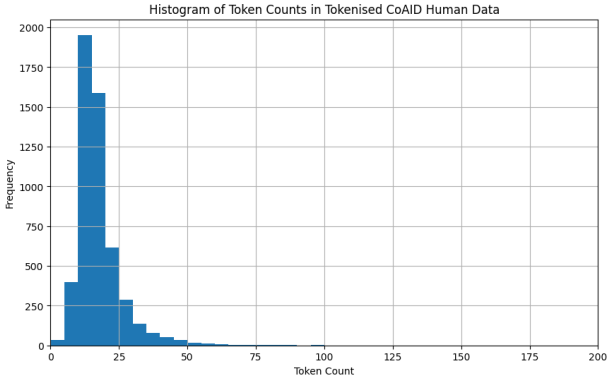


Figure 5.9: Human CoAID Tokenised Sequence Lengths.

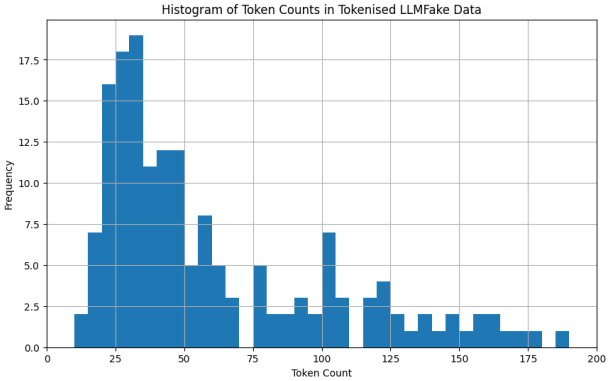


Figure 5.10: LLM CoAID Tokenised Sequence Lengths.

BERT can only take as input a maximum of 512 tokens, however with our GPU constraints we reduce this further by truncating (and padding where needed) all input sequences to 256 tokens to increase model efficiency. We are therefore careful to analyse the length of tokenised sequences before truncation so that we can evaluate in the discussion section how truncation affects model accuracy. In simple terms, if we have a long text which is tokenised into 1000 tokens and we then truncated it down to 256 tokens, we lose a lot of information and we may expect the model to be more likely to misclassify such an example compared with a shorter text.

5.5.4 Model and Hyper-Parameter Selection

In order to train an accurate model, we must ensure that we are choosing near-optimal hyper-parameters for training as well a good network architecture. We do this using a search grid over validation dataset to avoid data poisoning.

We select a model and hyper-parameters based on:

- **Frozen Parameters:** We choose between 2 common methods of fine-tuning BERT:
  - Freeze BERT body: here we freeze all weights from the original BERT involved in outputting the CLS token, and only update the parameters in the new classification head. This approach can be good on small datasets, where there is potential to overfit the model and cause catastrophic forgetting (where we lose the ability to identify important patterns that the original BERT had learned to identify).
  - Freeze no part of the model: here we allow both the new classification head and original BERT body to have their parameters updated. As long as we avoid overfitting this can result in a model which is more adapted to the specific task.
- **Learning Rates:** We look to find an optimal learning rate that rapidly converges yet does not overshoot the minima of the loss function.
- **Batch Sizes:** Larger batches provide more stability for gradient estimates whereas smaller batches can lead to faster convergence.
- **Number of Epochs:** This affects how many steps the model trains for. It is important to train the model as much as possible before overfitting starts to occur.

The original BERT authors recommend the following hyper-parameters:

- Batch size: 8, 16, 32, 64 or 128
- Learning rate: 1 to 5 e-5
- Epochs: 3 to 5

We perform a search grid over a selection of these recommendations, not using all possibilities due to GPU constraints. In this case, the hyper-parameter selection would take too long to run. Instead we just use the following search grid:

- **Learning Rates:** $\{1 \times 10^{-5}, 5 \times 10^{-5}\}$
- **Batch Sizes:** $\{16\}$
- **Number of Epochs:** $\{2, 4\}$
- **Model Initialization:** {"fully unfrozen", "body frozen"}

Dataset	Model Initialization	Learning Rate	Batch Size	Number of Epochs
Human PolitiFact	Fully Unfrozen	5e-5	16	2
Human GossipCop	Fully Unfrozen	1e-5	16	2
Human CoAID	Fully Unfrozen	5e-5	16	2
Human and LLMFake PolitiFact	Fully Unfrozen	1e-5	16	2
Human and LLMFake GossipCop	Fully Unfrozen	1e-5	16	2
Human and LLMFake CoAID	Fully Unfrozen	5e-5	16	2

Table 5.1: Optimal Model Configuration and Hyper-Parameters for BERT Models Trained on Different Datasets.

Examples of learning curves are demonstrated below. Note how we stop learning just as validation loss plateaus (before overfitting can begin).

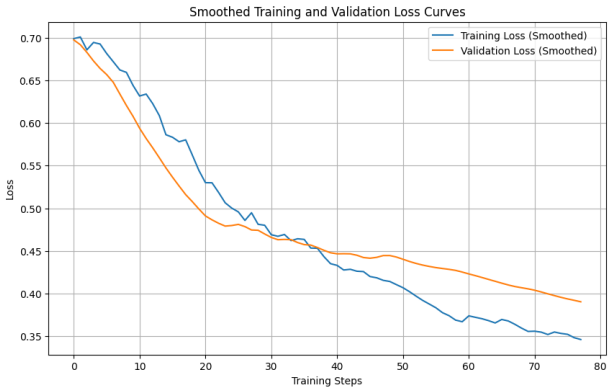


Figure 5.11: BERT Fine-Tuned on Human PolitiFact Data Learning Curve.

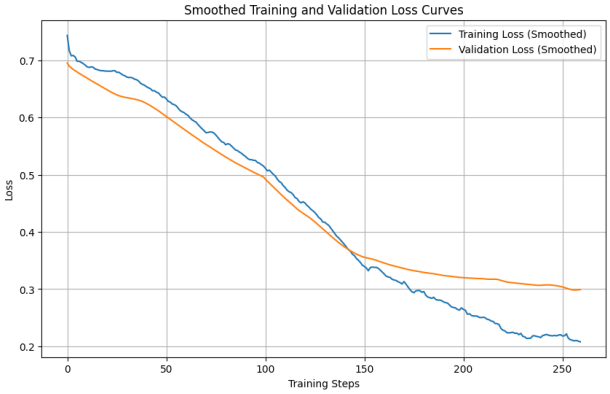


Figure 5.12: BERT Fine-Tuned on Human and LLM PolitiFact Data Learning Curve.

5.6 Performance Metrics

The LLMFake datasets contain only misinformation datapoints. Therefore, metrics such as precision, recall, and F1 score make no sense in this situation as they rely on the presence of true positives, true negatives, false positives and false negatives. Since the primary focus of this paper is on comparing detection difficulty between human and LLM-generated misinformation, our main performance metric is misinformation detection success rate. This is also the metric [3] Chen et al. use when evaluating detectors on similar datasets.

We also evaluate macro-averaged F1 score for detection performance on the human-generated datasets which contain both true information and misinformation datapoints. We choose macro-averaged F1 score for 2 reasons:

- F1 score combines precision and recall, which are dependent on true positives, false positives, and false negatives. This provides a more holistic view of model performance compared to just the misinformation detection success rate. Specifically, the F1 score can help indicate whether the model is sacrificing performance by generating many false positives to achieve a high misinformation detection success rate.
- Macro-averaging considers class imbalances by ensuring that the model's performance on the minority class is given equal weight as the majority class. This is the metric that [20] Wu et al. record on similar experiments.

We provide additional performance metrics such as precision, recall and accuracy in the appendix. However we only perform statistical significance tests for our main metrics: F1 score and misinformation detection success rate.

5.7 Choice of Statistical Significance Tests

We use statistical hypothesis tests to help us answer two key questions:

- Is LLM-generated misinformation harder to detect for detectors than human-generated misinformation?
- Is BERT fine-tuned on LLM-generated and human-generated data better at detecting misinformation than BERT fine-tuned solely on human-generated data?

5.7.1 The Correct Hypothesis Testing Method

Choice of statistical testing varies with many papers, even in reputable journals, reporting incorrect testing strategies ([6] Dror et al. 2018).

[16] Rainio et al., 2024 find that when comparing performance of two or more models, it is necessary to perform the tests multiple times to obtain multiple instances of the performance metric (ie. F1 score). In the binary classification setting we have to evaluate our models on several test sets to obtain enough values from other evaluation metrics for statistical testing. While the test sets should ideally come from fully different data sets, sometimes our only option is to use a resampling procedure to create multiple test sets from the same data. In practice, we must re-initialize, train, and test the models for several times and save the values of the evaluation metrics from the predictions of the test set on each iteration round. We should use the same training and test set for all the models on the same iteration round but vary them between the rounds because, otherwise, our conclusions about a potential difference between the models might be misled by some unknown factor in these specific data sets.

The choice of the exact test depends the task of the models, the evaluation metric used, and the number of test sets available. As some metrics produce only one value from a single test set and there might be only one data set, some type of resampling, such as repeated cross-validation, is often necessary. Because of this, the well-known tests such the paired t-test underestimate variance and do not produce reliable results. Instead, the use of non-parametric tests such as the Wilcoxon signed-rank test is recommended. However, the t-test is not recommended for this situation because it is strongly affected by outliers and not valid when resampled test sets are used.

5.7.2 What similar works have tested

- The SheepDog paper ([20] Wu et al., 2023) uses the Wilcoxon signed rank test (a non-parametric version of paired t-test which doesn't assume normality).
- [3] Chen et al., 2024 use the paired t test. Presumably they assume the data follows a normal distribution via the central limit theorem, however as mentioned previously this method is not appropriate when resampling data.

5.7.3 Our Statistical Hypothesis Tests

We follow the recommendation of [16] Rainio et al to resample performance metrics multiple times before using a Wilcoxon rank sign test. This is also what the Sheepdog paper did, to answer similar hypothesises.

Due to GPU constraints, we only resample performance metrics 5 times. When we then run a Wilcoxon rank sign test (and paired t-test for comparison out of interest), the p-value of the Wilcoxon rank sign test is much higher (less significant) than the paired t-test. We hypothesise that, if we had more compute power and could resample the performance metrics more times then the increased number of data points would decrease variability in the resampled performance metrics and therefore we would obtain more significant p-values from the Wilcoxon rank sign test.

CHAPTER 6

Results

RESULTS TABLES ARE PROVIDED HERE. SEE NEXT CHAPTER FOR DISCUSSION AND INTERPRETATION OF RESULTS. ADDITIONAL RESULTS IN APPENDIX.

6.1 PolitiFact

Model	Success Rate	Macro F1
Human BERT	89.47%	0.88
Human NB	92.11%	0.80
0-SHOT PHI3	76.27%	0.76
Human & LLM BERT	**93.51%**	**0.88**

Table 6.1: Model Performance on Human PolitiFact Dataset. Statistical significance testing on SR and macro F1 to see whether human and LLM BERT is better performing than the best baseline (other) model: (*) 10%.

(a) Part 1

Model	Overall Accuracy	Hallucination Content	Ambiguous Content	False Context	Incomplete Content	Outdated Content	Rewritten	Arbitrary Generation
Human BERT	66.19%	45.0%	17.24%	40%	34.48%	34.48%	92.59%	100%
Human NB	71.70%	70%	44.83%	53.33%	41.38%	41.38%	90.74%	85%
0-SHOT PHI3	66.91%	25.0%	59.09%	61.90%	47.62%	86.36%	69.05%	75%
Human & LLM BERT	**95.44%***	**100%***	**79.3%***	**96.67%***	**89.66%***	**96.56%**	**98.15%***	**100%***

(b) Part 2

Model	Total Fabrication	Unsubstantiated Content	Open Generation	Paraphrase	Health Misinformation	Politics Misinformation
Human BERT	44.83%	37.93%	90.74%	87.04%	100%	100%
Human NB	51.72%	41.38%	88.89%	94.44%	100%	100%
0-SHOT PHI3	72.22%	75.00%	43.75%	70.27%	100%	100%
Human & LLM BERT	**89.66%***	**93.10%***	**98.15%***	**98.15%***	**100%**	**100%**

Table 6.2: Model Success Rate on LLMFake PolitiFact Dataset. Statistical significance testing on SR across LLM-generation categories to see whether Human and LLM BERT is better performing than the best baseline (other) model: (*) 10%.

6.2 GossipCop

Model	Success Rate	Macro F1
Human BERT	73.77%	0.80
Human NB	**74.92%**	0.75
0-SHOT PHI3	45.63%	0.62
Human & LLM BERT	68.65%	**0.82***

Table 6.3: Model Performance on Human GossipCop Dataset. Statistical significance testing on SR and Macro F1 to see whether Human and LLM BERT is better performing than the best baseline (other) model: (*) 10%.

Model	Overall	Open Generation	Paraphrase	Rewritten
Human BERT	76.98%	69.57%	76.09%	85.11%
Human NB	72.66%	67.39%	73.91%	76.60%
0-SHOT PHI3	31.73%	32.14%	32.43%	30.77%
Human & LLM BERT	**97.84%***	**100%***	**95.65%***	**97.87%***

Table 6.4: Model Success Rate on LLMFake GossipCop Dataset. Statistical significance testing on SR across LLM-generation categories to see whether Human and LLM BERT is better performing than the best baseline (other) model: (*) 10%.

6.3 CoAID

Model	Success Rate	Macro F1
Human BERT	**98.91%**	0.97
Human NB	62.30%	0.82
0-SHOT PHI3	89.80%	0.51
Human & LLM BERT	98.36%	**0.98***

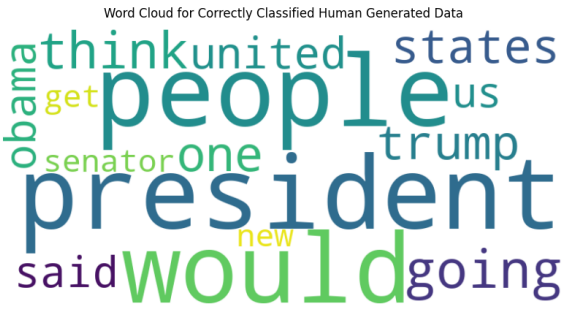
Table 6.5: Model Performance on Human CoAID Dataset. Statistical significance testing on SR and Macro F1 to see whether Human and LLM BERT is better performing than the best baseline (other) model: (*) 10%.

Model	Overall	Open Generation	Paraphrase	Rewritten
Human BERT	98.08%	100.00%	94.12%	100.00%
Human NB	46.15%	11.76%	58.82%	66.67%
0-SHOT PHI3	82.35%	50.00%	92.86%	91.67%
Human & LLM BERT	**100%**	**100%**	**100%**	**100%**

Table 6.6: Model Success Rate on LLMFake CoAID Dataset. Statistical significance testing on SR across LLM-generation categories to see whether Human and LLM BERT is better performing than the best baseline (other) model: (*) 10%.

6.4 Understanding Misclassifications

6.4.1 Manual Inspection



(a) Word Cloud of Correctly Classified Texts on Human PolitiFact via Human BERT.

(b) Word Cloud of Inorrectly Classified Texts on Human PolitiFact via Human BERT.

Figure 6.1: Wordclouds for Correctly and Incorrectly Classified Human-Generated PolitiFact Texts.

We attempt to analyse why certain texts are misclassified by models and other texts are classified correctly. At first we conduct a manual review of texts to try and identify any patterns but nothing obvious presents itself as different between the correctly and incorrectly classified datapoints in terms of style or word usage. Additionally we look at the word clouds, so that we can get a view of just the most frequent words. Figures in this section demonstrate our analysis process focusing on the human PolitiFact dataset. The word clouds do not give us any additional insight. Indeed, the words used across correctly and incorrectly classified texts are very similar, with "president", "people", "said" and other words appearing in both.

6.4.2 Model Confidence on Misclassifications

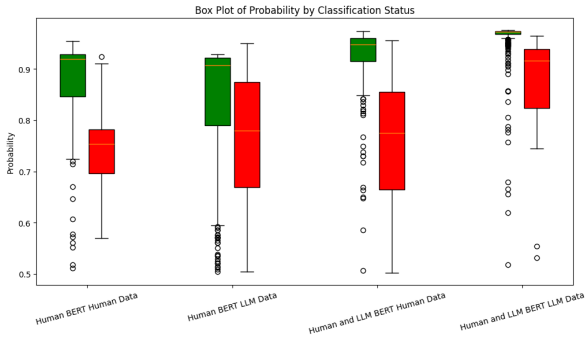


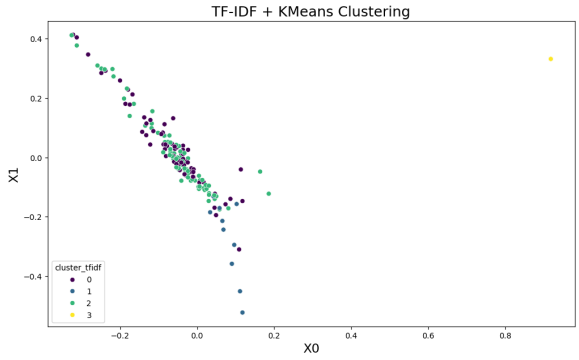
Figure 6.2: Model Confidence Scores on Correct and Incorrect Classifications on PolitiFact Data.

As expected, the model is less confident (assigns lower probabilities) to texts which it then misclassifies. This suggests that the model does have some understanding of which texts are hard to classify. We therefore considered an approach where we post-process predictions to set any predictions with a low confidence (near 0.5) to a 3rd class "unsure". In real world usage, i.e. for social media fake news detection, this could then allow a person to make the final classification. However, there are 2 key problems with this approach. The first is that there would be far too many misclassifications for human labour to realistically handle. The second being that there are also many correctly classified points with low confidence, as such we would be removing these correct predictions and duplicating work.

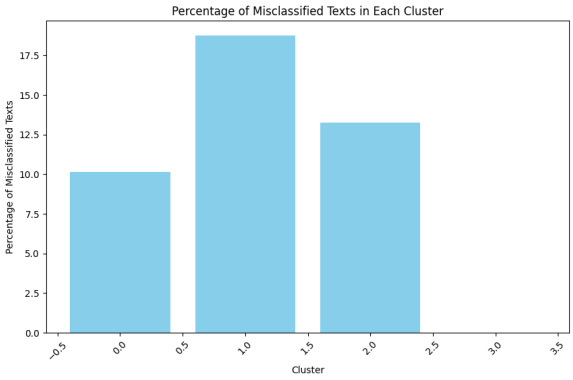
6.4.3 Naive Clustering

As an interesting sub-analysis, we attempt to cluster the human BERT predictions for the human PolitiFact data using KNN. Here, we look to see whether groups of texts close in latent space have higher misclassification rates than others. If this was the case, then we could look to identify the features which

make these texts neighbours to gain insight into the misclassification behaviour. However, we do not find there to be a significant difference in misclassification rates across clusters so this analysis was not helpful.



(a) PCA Dimensionality Reduced KNN Clusters on PolitiFact Human Data Predictions Via Human BERT.



(b) Cluster Misclassification Rates on PolitiFact Human Data Predictions Via Human BERT.

Figure 6.3: KNN Clustering of Predictions.

6.5 Detection Hardness for Human BERT on Human vs LLM Data

Dataset	p-value (Wilcoxon)	p-value (Paired t)
PolitiFact	0.0625	0.0012
GossipCop	0.0679	0.0573
CoAID	0.0625	0.0012

Table 6.7: Statistical P-Values for Wilcoxon Rank Sign Test and Paired T-Test Comparing Human BERT Detection Difficulty on Human-Generated vs LLM-Generated Data.

CHAPTER 7

Discussion

7.1 The Challenge of Testing Statistical Significance

	alpha values						
n	0.001	0.005	0.01	0.025	0.05	0.10	0.20
5	--	--	--	--	--	0	2
6	--	--	--	--	0	2	3
7	--	--	--	0	2	3	5
8	--	--	0	2	3	5	8
9	--	0	1	3	5	8	10
10	--	1	3	5	8	10	14
11	0	3	5	8	10	13	17
12	1	5	7	10	13	17	21
13	2	7	9	13	17	21	26
14	4	9	12	17	21	25	31
15	6	12	15	20	25	30	36
16	8	15	19	25	29	35	42
17	11	19	23	29	34	41	48
18	14	23	27	34	40	47	55
19	18	27	32	39	46	53	62
20	21	32	37	45	52	60	69
21	25	37	42	51	58	67	77
22	30	42	48	57	65	75	86
23	35	48	54	64	73	83	94
24	40	54	61	72	81	91	104
25	45	60	68	79	89	100	113
26	51	67	75	87	98	110	124
27	57	74	83	96	107	119	134

	alpha values						
n	0.001	0.005	0.01	0.025	0.05	0.10	0.20
28	64	82	91	105	116	130	145
29	71	90	100	114	126	140	157
30	78	98	109	124	137	151	169
31	86	107	118	134	147	163	181
32	94	116	128	144	159	175	194
33	102	126	138	155	170	187	207
34	111	136	148	167	182	200	221
35	120	146	159	178	195	213	235
36	130	157	171	191	208	227	250
37	140	168	182	203	221	241	265
38	150	180	194	216	235	256	281
39	161	192	207	230	249	271	297
40	172	204	220	244	264	286	313
41	183	217	233	258	279	302	330
42	195	230	247	273	294	319	348
43	207	244	261	288	310	336	365
44	220	258	276	303	327	353	384
45	233	272	291	319	343	371	402
46	246	287	307	336	361	389	422
47	260	302	322	353	378	407	441
48	274	318	339	370	396	426	462
49	289	334	355	388	415	446	482
50	304	350	373	406	434	466	503

Figure 7.1: Wilcoxon Rank Sign Test Statistics.

The Wilcoxon rank sign test is a non-parametric test which determines if the difference between two sets of values, such as performance metrics, is non-zero. We obtain these performance metrics by resampling: retraining the model and obtaining new sets of predictions. Due to our GPU constraints, we only resample performance metrics 5 times. Having such a small sample size of performance metrics, means that the estimated variance of the sample is less stable. Additionally, statistical power is reduced. **These issues mean that it is not possible to test 5% significance on such a sample size (as shown in figure 7.1). Therefore, we only test at the 10% significance level.**

However, we hypothesise that many of our results are likely significant at the 5% and potentially 1% levels too, due to the values of test statistics observed and consistent large differences in our findings across several experiments. We hope to use a more powerful GPU in future to re-run these experiments a few more times to be able to test this hypothesis and obtain more meaningful significance tests.

7.2 Overview of Key Findings

7.2.1 Fine-Tuning BERT with LLM-Generated Data Augmentation Can achieve SOTA Performance on Human-Generated Misinformation Detection

We demonstrate that BERT fine-tuned on human-generated and LLM-generated data is the best performing model (in terms of macro F1 score) at detecting human-generated misinformation across all 3 human datasets. Across GossipCop and CoAID, the results are statistically significant. Here we even surpass the performance of BERT fine-tuned specifically, and only, on human-generated data. This suggests that using LLMs for data augmentation, can increase the model performance for human-generated misinformation detection.

We conduct a comparison study by looking at results reported by [20] Wu et al, 2023., who train SOTA misinformation detectors on the human PolitiFact and GossipCop datasets. In their work they propose the SheepDog model which achieves best performance. Our model equals SheepDog performance on the PolitiFact datatset and surpasses SheepDog on the GossipCop dataset.

Model	PolitiFact	GossipCop
SheepDog	**0.88**	0.76
Our Model	**0.88**	**0.82**

Table 7.1: Comparison of F1 Scores Reported on Human-Generated Datasets Between Our Paper and the SheepDOG Paper.

7.2.2 Fine-Tuning BERT with LLM-Generated Data Augmentation Achieves SOTA Results on LLM-Generated Misinformation Detection

BERT fine-tuned on human-generated and LLM-generated data either matches or outperforms our other models on LLM-generated misinformation detection success rate across every dataset and every generation category. This demonstrates that fine-tuning models on human-generated data is insufficient to make the model robust to LLM empowered attacks. A key consideration here, is how our model would perform on LLM-generated data of an unseen category (out of distribution data). Therefore, it would be interesting to investigate fine-tuning the model on certain misinformation generation categories and seeing how it can generalise to unseen categories in future.

We conduct a comparison study by looking at results reported by [3] Chen et al, 2024., who create the LLMFake dataset and record results of 0-shot LLM detectors. Fine-tuned language models have frequently been shown to be more successful at misinformation detection than 0-shot LLMs, therefore it is no surprise that our model sets new SOTA performance metrics on the dataset.

Data	Paraphrase		Rewriting		Open-Ended	
	GPT-4	Our Model	GPT-4	Our Model	GPT-4	Our Model
PolitiFact	56.0%	**98.2%**	53.6%	**98.2%**	41.6%	**98.2%**
GossipCop	30.0%	**95.7%**	25.0%	**97.9%**	25.7%	**100.0%**
CoAID	82.2%	**100.0%**	73.3%	**100%**	52.7%	**100.0%**

Table 7.2: LLMFake Misinformation Detection Success Rates.

7.2.3 Insufficient Evidence to Suggest Current Misinformation Detectors Are Worse At Detecting LLM-Generated Misinformation than Human-Generated Misinformation

We hypothesise that the standard approach of training misinformation detection on human-generated datasets creates models that may not generalise well to LLM-generated data. Infact across GossipCop and CoAID datasets, the opposite is true and we observe a slight increase in success rate on the LLMFake test set. On the PolitiFact dataset, there is a significant drop off in success rate on the LLMFake data.

Since we have contradicting and inconclusive results, more investigation is needed here. These mixed results may stem from the small GossipCop and CoAID LLMFake data sizes.

7.2.4 Truncating BERT Input Sequences May Decrease Model Accurracy

We observe that human-generated and LLM-generated CoAID data contains the shortest input sequences, which do not need any truncation as they are (mostly) below the token limit we set for our BERT and Phi3 models as shown in chapter 5. On the other hand, GossipCop and PolitiFact have many sequences above the truncation limit. As perhaps expected, our BERT models achieve their best performance metrics on the CoAID data leading us to hypothesis that this may be due to the loss of information from truncation in BERT models on the other datasets. Alternatively, it may be that shorter input sequences are just simpler and have less text to confuse our models.

7.2.5 Naive Bayes and 0-shot Phi3 Observe Highly Volatile Performance

Naive Bayes observes overall misinformation detection success rates from 46.15% to 92.11% whilst Phi3 success rates range from 31.73% to 89.80%. On the contrary, BERT models (especially when fine-tuned with LLM data augmentation) show much more consistency. This is likely due to BERT making use of the broader language patterns which transformer-based language models are able to spot.

Whilst NB does not perform badly, we note that it uses the proportions of class imbalances as a feature since it calculates prior probability. This contrasts the approach in BERT where we handle class imbalances to mitigate any bias in the model towards predicting either real or fake classifications. This may mean that Naive Bayes generalises poorly to datasets with different proportions of real to fake articles.

7.3 Limitations and Future Work

7.3.1 Sub-Optimal Models

A limitation of this work stems from the workarounds implemented to deal with having insufficient GPU resources. We expect the loss of information from truncating tokenised input sequences to severely hinder model performance on classifying longer texts. Additionally, having to use a small search grid for hyper-parameter selection may have lead to sub-optimal model training conditions. However, the fact that our LLM-augmented fine-tuned BERT model performs well, surpassing recent SOTA detectors on various datasets, in spite of these limitations, demonstrates the strengths of our methodology. We would like to re-run our code with a more powerful and flexible GPU to:

- Fine-tune our models in optimal conditions (minimal truncation and large hyper-parameter search grids).
- Run a larger number of (and more computationally-expensive) SOTA detectors in the same environment as our model to compare all models in the most accurate way possible, accounting for different python libraries having different computational methods. This would allow us to conclusively confirm whether our model out-performs i.e. the SheepDog model. We would consider models such as GPT-4 and improved BERT variations.

7.3.2 Low Power Statistical Tests

Our work only tests statistical significance at the 10% level since we only resample performance metrics 5 times. This makes it hard for us to draw conclusive evidence that certain models are indeed superior to others. The addition of a more powerful GPU would allow us to resample performance metrics more times, potentially decreasing the variability in the resampled performance metrics, and increasing statistical power. Therefore, we would hope to be able to highlight the significant improvements of our LLM-augmented fine-tuned BERT model compared to all other baselines.

7.3.3 Explainability

In future work, we may also wish to test a model which extends beyond binary classification to make use of the LLM-generation technique labels in the LLMFake data. Having the model output why it believes a text is false would enhance the explainability of the model, potentially allowing for it to be used in more sensitive settings such as healthcare content, where understanding how a model reaches conclusion is of great importance. It would be interesting to see how much overall performance the model would trade to gain this functionality.

7.3.4 Generalisability

Furthermore, we wish to test in future the ability of our models to generalise to completely out of distribution data. For example, we wish to setup an experiment where we train our models on the PolitiFact datasets and test how well they can generalise to the GossipCop datasets. This may or may not highlight some weaknesses in our models which would likely then be observed in a real world setting where the models would have to generalise across many domains.

[10] Liu et al., 2022, state that since reliable labeling of massive datasets for various application domains is often expensive and prohibitive, for a task without sufficient labeled datasets in a target domain, we wish to be able to train a model in one domain and then use it on another. This learning strategy, however, suffers from shifts in data distributions, i.e. domain shift. In future, we would look to handle domain shift using unsupervised domain adaptation, a transfer learning method where we take a model which has been fine-tuned to detect misinformation on a specific source domain (i.e. on PolitiFact) and leverage unsupervised data to map the model distribution onto a target domain (i.e. GossipCop).

7.3.5 Improving Large Language Model Detectors

Similarly to the process of fine-tuning language models such as BERT for a specific task, we may further train LLMs for misinformation detection in a few-shot setting. This would provide us with a more direct and fair comparison between generative LLM and task-specific language model performance, allowing us to directly compare the strengths and weaknesses of both architectures in how they use either causal or bidirectional attention mechanisms. Further training LLMs is a resource intensive process and therefore we would again require a more powerful GPU.

7.4 Conclusion

Fine-tuning language models, such as BERT, on LLM-generated data as a form of data augmentation can improve misinformation detection performance to state-of-the-art levels on both human-generated and LLM-generated test data. This highlights a way forward of using an LLM data augmentation step in model training which could potentially be utilised in not only misinformation detection but other forms of text classification too such as sentiment analysis or spam detection. This study, demonstrates the promise of such methodologies and now requires further investigation to conclusively prove state-of-the-art status and find optimal model setups to improve performance further.

CHAPTER 8

Appendix

8.1 Full Results Tables

Here we display the full results tables from our experiments, including performance metrics such as precision, recall and accuracy. We did not display these metrics in the body of our paper since we chose success rates and macro f1 score as our main metrics.

8.1.1 PolitiFact

Model	Success Rate	Macro F1	Class 0 Precision	Class 0 Recall	Class 1 Precision	Class 1 Recall	Accuracy
Human BERT	89.47%	0.88	0.84	0.89	**0.92**	0.87	0.88
Human NB	92.11%	0.80	0.71	0.92	0.92	0.71	0.80
0-SHOT PHI3	76.27%	0.76	0.71	0.76	0.81	0.77	0.77
Human & LLM BERT	**93.51%**	**0.88**	**0.98**	**0.94**	0.74	**0.89**	**0.93**

Table 8.1: Model Performance on Human PolitiFact Dataset. Statistical significance testing on SR and Macro F1 to see whether Human and LLM BERT is better performing than the best baseline (other) model: (*) 10%.

8.1.2 GossipCop

Model	Success Rate	Macro F1	Class 0 Precision	Class 0 Recall	Class 1 Precision	Class 1 Recall	Accuracy
Human BERT	73.77%	0.80	0.66	0.74	0.91	0.88	0.85
Human NB	**74.92%**	0.75	0.55	0.75	**0.91**	0.81	0.80
0-SHOT PHI3	45.63%	0.62	0.40	0.46	0.83	0.79	0.71
Human & LLM BERT	68.65%	**0.82***	**0.75**	0.72	0.90	**0.91**	**0.86**

Table 8.2: Model Performance on Human GossipCop Dataset. Statistical significance testing on SR and Macro F1 to see whether Human and LLM BERT is better performing than the best baseline (other) model: (*) 10%.

8.1.3 CoAID

Model	Success Rate	Macro F1	Class 0 Precision	Class 0 Recall	Class 1 Precision	Class 1 Recall	Accuracy
Human BERT	**98.91%**	0.97	0.91	0.99	1.00	0.98	0.98
Human NB	62.30%	0.82	0.77	0.62	0.92	0.96	0.90
0-SHOT PHI3	89.80%	0.51	0.17	0.90	0.98	0.58	0.61
Human & LLM BERT	98.36%	**0.98***	**0.95**	**0.99**	**1.00**	**0.99**	**0.99**

Table 8.3: Model Performance on Human CoAID Dataset.Statistical significance testing on SR and Macro F1 to see whether Human and LLM BERT is better performing than the best baseline (other) model: (*) 10%.

8.2 Code Structure

Find our code at www.github.com/JacobShort11/adapting_misinformation_detectors_paper

Alternatively university personnel may access the code & data through the university GitLab server.

Here is a description of each code file:

- **data_preparation.ipynb:** data cleaning and exploration.

- **bert_politifact.ipynb:** training, validation, testing and resampling BERT models on human & LLMFake PolitiFact datasets.
- **bert_gossipcop.ipynb:** training, validation, testing and resampling BERT models on human & LLMFake GossipCop datasets.
- **bert_coaid.ipynb:** training, validation, testing and resampling BERT models on human & LLMFake CoAID datasets.

- **phi_politifact.ipynb:** testing PHI3 on human & LLMFake PolitiFact datasets.
- **phi_gossipcop.ipynb:** testing PHI3 on human & LLMFake GossipCop datasets.
- **phi_coaid.ipynb:** testing PHI3 on human & LLMFake CoAID datasets.

- **nb_politifact.ipynb:** training and testing NB on human & LLMFake PolitiFact datasets.
- **nb_gossipcop.ipynb:** training and testing NB on human & LLMFake GossipCop datasets.
- **nb_coaid.ipynb:** training and testing NB on human & LLMFake CoAID datasets.

- **misclassification_analysis.ipynb:** analysis into why models are misclassifying certain texts.

- **statistical_tests.ipynb:** Wilcoxon rank sign tests on performance metrics.

Please note: you must independently download the relevant PolitiFact, GossipCop, CoAID and LLMFake datasets.

Bibliography

[1] Sara Abdali, Sina Shaham, and Bhaskar Krishnamachari. Multi-modal misinformation detection: Approaches, challenges and opportunities. *arXiv preprint*, 2022. URL: https://arxiv.org/abs/2203.13883.

[2] Tom Brown, Benjamin Mann, and Nick Ryder. Language models are few-shot learners. *NeurIPS*, 2020. doi:10.48550/arXiv.2005.14165.

[3] Canyu Chen and Kai Shu. Can llm-generated misinformation be detected? *ICLR*, 2024. doi:10.48550/arXiv.2309.13788.

[4] Abhimanyu Deubey and Abhinav Jauvri et al. The llama 3 herd of models. *Meta*, 2024. doi:10.48550/arXiv.2407.21783.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *ACL*, pages 4171–4186, 2019. doi:10.18653/v1/N19-1423.

[6] Rotem Dror, Segev Shlomov, and Roi Reichart. The hitchhiker's guide to testing statistical significance in natural language processing. *ACL*, 56.1:1383–1392, 2018.

[7] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. *ICLR*, 2021. doi:10.48550/arXiv.2006.03654.

[8] Sepp Hochreiter. Long short-term memory. *Neural Computation*, 1997. URL: https://www.bioinf.jku.at/publications/older/2604.pdf.

[9] Kung-Hsiang Huang, Kathleen McKeown, and Preslav Nakov. Faking fake news for real fake news detection: Propaganda-loaded training data generation. *ACL*, 61:14571–14589, 2023. doi:10.18653/v1/2023.acl-long.815.

[10] Xiaofeng Liu and Chaehwa Yoo. Deep unsupervised domain adaptation: A review of recent advances and perspectives. *APSIPA*, 2022. doi:10.1561/116.00000192.

[11] Yinhan Liu and Myle Ott. Roberta: A robustly optimized bert pretraining approach. *Arxiv*, 2019. doi:10.48550/arXiv.1907.11692.

[12] Shervan Minaee and Nal Kalchbrenner. Deep learning-based text classification: A comprehensive review. *ACM*, 54(3):1–40, 2021. doi:10.1145/3439726.

[13] Veronica Perez-Rosas and Bennett Kleinberg. Automatic detection of fake news. *ACL*, 27:3391–3401, 2018. URL: https://aclanthology.org/C18-1287.

[14] Fabio Petroni and Tim Rocktaschel. Language models as knowledge bases? *ACL*, 2019. doi:10.18653/v1/D19-1250.

[15] Huyen Phan, Ngoc Nguyen, and Dosam Hwang. Fake news detection: A survey of graph neural network methods. *Applied Soft Computing*, 2023. doi:10.1016/j.asoc.2023.110235.

[16] Oona Rainio, Jarmo Teuho, and Riku Klen. Evaluation metrics and statistical tests for machine learning. *Nature*, 14:6086, 2024. URL: https://www.nature.com/articles/s41598-024-56706-x.

[17] Mike Schuster and Kaisuke Nakajima. Japanese and korean voice search. *IEEE*, 2012. doi:10.1109/ICASSP.2012.6289079.

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, and Jakob Uszkoreit. Attention is all you need. *NIPS*, 2017. doi:10.48550/arXiv.1706.03762.

[19] Jason Wei and Yi Tay. Emergent abilities of large language models. *Transactions on ML Research*, 2022. doi:10.48550/arXiv.2206.07682.

[20] J Wu and B Hooi. Fake news in sheep's clothing: Robust fake news detection against llm-empowered style attacks. *CoRR*, 2023. doi:10.48550/arXiv.2310.10830.

[21] Qizhe Xie and Zihang Dai. Unsupervised data augmentation for consistency training. *NeurIPS*, 2019. doi:10.48550/arXiv.1904.12848.

[22] Xuan Zhang and Wei Gao. Towards llm-based fact verification on news claims with a hierarchical step-by-step prompting method. *ACL*, 13:996–1011, 2023. doi:10.18653/v1/2023.ijcnlp-main.64.