Investigating the Adversarial Case in Misinformation Detection

KRDX6[1]

Master of Science in Machine Learning

Supervised by Emine Yilmaz and Qiang Zhang

Submission date: September 13, 2021

[1]**Disclaimer:** This report is submitted as part requirement for the Master of Science in Machine Learning at UCL. It is substantially the result of my own work except where explicitly indicated in the text.
The report will be distributed to the internal and external examiners, but thereafter may not be copied or distrbuted except with permission from the author.

Abstract

Misinformation has been a permanent feature of the internet since its inception, yet in recent years, its rising ubiquity on social networks has added urgency to the issue of combating it. Swathes of human fact-checkers are tasked with policing content but cannot match the endless flows of information, as a result, research into automated solutions has been pursued for some number of years. As solutions are still being researched, the potential impact of real-world users on such models is relatively unknown and an attempt to elucidate this is the crux of this thesis. This work assumes that real-world users may behave maliciously and, by analogy to the field of Adversarial Machine Learning, seeks to understand this effect. This work contributes a novel method, denoted as 'Targeted-Importance Scores' (or 'T-Scores') for understanding the globally most important features to a model's decision making. T-Scores are utilised in a novel, black-box, genetically-inspired adversarial attack, denoted as the `Misinformer` algorithm, that attempts to perturb an input in order to convince an attacked model that the input is in fact truthful. The `Misinformer` is shown to reduce the accuracy of the best-performing models developed, in this work, from $\sim$75% to $\sim$46% which is the frequency of the targeted class; in other words, the best models are reduced to the performance of simply classifying true for all data points. Finally, the `Misinformer` is shown to be resistant to the most-popular defense algorithm from the Adversarial Machine Learning literature.

Contents

1 Introduction **2**

1.1 Motivation . . . 2

1.2 Objectives . . . 3

1.3 Structure of the Thesis . . . 4

2 Background and related work **5**

2.1 Natural Language Processing (NLP) . . . 6

2.1.1 Representing Text . . . 6

2.1.2 Paraphrasing Models . . . 14

2.2 Adversarial Machine Learning . . . 15

2.2.1 Attacks . . . 16

2.2.2 Defenses . . . 20

2.3 Explainable Artificial Intelligence (XAI) . . . 20

2.3.1 Overview . . . 20

2.3.2 LIME . . . 21

2.4 Misinformation Detection . . . 23

2.4.1 Dataset . . . 24

2.4.2 Previous approaches for the Pheme dataset . . . 25

3 Methods **28**

3.1 Model Design . . . 29

3.2 Adversary . . . 31

3.2.1 Adversarial Domain Definition . . . 31

3.2.2 Extending Global Homogeneity-Weighted Importance . . . 32

3.2.3 Constituent attacks . . . 34

3.2.4 Fixed Adversary . . . 36

3.2.5 Genetic Adversary . . . 38

4 Experiments **43**

4.1 Performance on the Pheme Dataset . . . 43

4.2 Model Intepretation 47
4.3 Adversarial Performance 48
4.3.1 Attack 48
4.3.2 Defense 53

5 Conclusion **54**
5.1 Summary 54
5.2 Future work 55

An anonymized repository with the code for this project can be found at: https://anonymous.4open.science/r/misinformer-56BB/

Chapter 1

Introduction

The advent of social media platforms, and the internet at large, has brought about unprecedented access to information; individuals have the ability to, at a whim, query the answer to any intriguing question. Such an endless flow of information has a multitude of intellectual and social benefits, however, when it is unchecked, can be imperceptibly poisoned with misinformation. The results are often catastrophic to the society in which they take place; research into misinformation on the social media platform Twitter has shown that falsehoods are 70% more likely to be retweeted and spread around six times faster than true statements [Vosoughi et al., 2018]. The real-world impact, at the time of writing, is obvious; in a recent random sample, 24.8% of tweets relating to the COVID-19 pandemic contained misinformation [Kouzy et al., 2020], the content of which might alter the behaviour of the reader and lead to wider public health risks. The problem of misinformation-detection can be approached from a machine learning perspective, in which it is attempted to design and train models that can distinguish content containing false information from the truth.

1.1 Motivation

The visibility and urgency of the problem of misinformation detection has provided enough incentive for artificial intelligence-based solutions to be researched [Islam et al., 2020, Jain et al., 2016, Lee et al., 2021, Su et al., 2020]; the vast majority of research is dedicated to showing that machine learning models may be designed that can detect falsehoods on social media platforms (commonly Twitter). Let us consider the event that such a model is released into the public domain; for example, that it is used to verify content as it is published onto a social network and content that is deemed false is removed. In such a case, the data input to the model can take a very different form; instead of a fixed, known and carefully-curated dataset the model is now subjected to users that can adapt their behaviour in order to manipulate the response of the model; this could mean that real-world performance may not closely align

with the performance observed on the benchmark. This use-case is commonly known as the adversarial domain in machine learning [Biggio et al., 2013, Szegedy et al., 2014, Goodfellow et al., 2015] and the potential effect of adversarial users on misinformation-detection is the subject of this thesis. The implicit hypothesis of any released misinformation model is that it can detect statements which are true, false or unverified from one another. However, all the data collected and used to benchmark such models is from users who are unaware of their participation in the modelling effort; in other words, these are users with entirely unadapted behaviour and the assumption is that their observed behaviour will more or less mirror that of the dataset used to train the model or that the model will be able adapt to the new behaviour.

In some sense, social media platforms and the internet in general, are predominantly fully observable, that is, a user subjected to the decision making of the model will be aware of the decision taken; for example, if the model decides that a particular tweet contains misinformation then the user may observe that the tweet does not appear on the Twitter platform or the platform may inform them directly, as is the case with hate-speech for example. Hence, it is required to assume that users will in-turn adapt their behaviour in order to, if possible, obtain the decision desired when interacting with such a model. In plainer terms, a malicious user will attempt to 'trick' the model into, for instance, misclassifying a false statement as being true or vice-versa. Furthermore, fact-checking, at present, is a laborious process that is carried out, predominantly, by human workers; automating such a process could lead to an increase in the fidelity of information on the internet and also reduce the load of such workers if a "human-in-the-loop"[1] approach is to be pursued. Regardless, the potential positive impact of accurate and, more importantly, robust misinformation-detection models is large.

1.2 Objectives

The motivation of the project stems from a desire to better understand the robustness of potential misinformation-detection models and this lends itself to a series of objectives and research questions that are considered to be the subject of this thesis.

1. There is a large body of research into misinformation-detection models, yet it is unknown as to what extent the models developed are resistant to adversarial attacks? This thesis develops a novel adversarial algorithm and records the impact on misinformation-detection models of our own design as a means of gauging this.

2. When a misinformation-detection model is trained, what precisely does it learn to classify? To what extent does the model rely on the prescence of individual words which can be manipulated?

[1] "Human-in-the-loop" typically refers to a framework where a model is used in collaboration with a human decision-maker to obtain increased accuracy in some problem domains [Zanzotto, 2017]

1.3 Structure of the Thesis

This chapter has outlined a brief and predominantly non-technical motivation underpinning this thesis. The second chapter is a summary of the literature upon which this thesis is founded; a plethora of relevant works in the fields of adversarial machine learning; natural language processing and explainable artificial intelligence, as applicable to this thesis, are reviewed. The dataset used as a benchmark, known as the Pheme dataset [Zubiaga et al., 2016] , is also introduced, with a justification provided as to its suitability for the research pursued. The third chapter outlines the approach of this thesis concretely; the literature summarised in the second chapter is amalgamated to form a novel adversarial attack algorithm with the aim of testing the robustness of misinformation-detection models. In the fourth chapter, the techniques developed are evaluated on the Pheme dataset with an emphasis on both performance on the benchmark as well as performance in the adversarial case. Finally, in the fifth chapter, this thesis is summarised and potential directions for future work are described.

Chapter 2

Background and related work

In this chapter, relevant literature and background knowledge are summarised in relation to this thesis; these works stem from a variety of fields within machine learning reseach. The task of misinformation detection is primarily performed in a text-based context; the field of research focused on the development of artificially intelligent programs for understanding text is commonly known as *Natural Language Processing (NLP)* and, as a result, this chapter begins with a brief summary of the relevant techniques within NLP.

In order to comprehend the robustness of misinformation-detection models, one must attempt to model, in some regard, the behaviour of a potentially malicious user. The field of research associated with such tasks, as well as the development of techniques to improve the robustness of models, is commonly known as *Adversarial Machine Learning*. A brief summary of the relevant techniques for both 'attacking'[1] and 'defending'[2] a model are described.

The justification of a misinformation-detection model in making its classification decision is critical to understanding its performance. For example, if the model simply correlates a particular class with a certain word, rather than identifying the honesty or dishonesty itself then it is clearly not robust; such models would be grossly fallible when subjected to real-world users. Understanding the decision-making of these models is to elucidate their failure cases and is an important step towards ensuring robustness. The field concerned with techniques for interpreting the decision-making of machine learning models is knowns as *Explainable Artificial Intelligence* and the relevant techniques from this field are summarised.

Adjacent works in the literature are outlined; machine learning approaches to misinformation-detection have been numerous in recent years and the field is growing as the wider issue of misinformation receives more attention. The existing literature guides the approach of this thesis and therefore a literature review is provided.

[1] The process of attempting to discover misclassifications inherent to a model
[2] The process of attempting to improve a model's resistance to nefarious inputs

2.1 Natural Language Processing (NLP)

The field of NLP has progressed from a core-set of techniques which are often referred to as the 'classical approaches'; such techniques form the foundation of recent developments and are outlined in order to aide in understanding some of the recent developments utilised.

2.1.1 Representing Text

Many machine learning techniques, from a simple regression to a complex deep neural-network, are challenging to formulate on non-numerical data. Suppose that one wishes to fit a logistic regressor to predict the sentiment of a piece of text, it is not immediately obvious how to represent, say, *'that movie was amazing'* in a numerical format; yet it is clear that a good representation may allow for a plethora of models to become suddenly applicable to NLP-tasks. This task, the transformation of text-based content into a numerical format is at the core of NLP and is reviewed from end-to-end.

Tokenization The first stage of representing text is to define the units that it comprises of, a process known as 'tokenization'. This is often done at the 'word-level', that is, 'the donkey leapt over the horse' is simply split into each of its constituent words (i.e. ['the', 'donkey', 'leapt', 'over', 'the', 'horse']) and those are considered to be the 'building blocks' of the sentence and should each be individually represented.

The Bag-of-Words Model

The bag-of-words model [Harris, 1954] is an approach that simplifies the notion of a sentence or document; instead of considering a sentence as an ordered sequence of words, the bag-of-words model represents a sentence as an unordered collection of words each of which can appear multiple times. This relies on the notion of a multiset (or 'bag') which is an extension of the mathematical notion of a set without the constraint that each element cannot occur more than once.

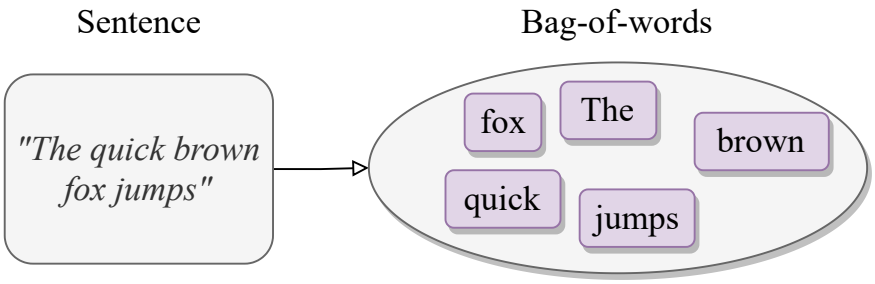

Figure 2.1: A diagram exemplifying the bag-of-words model applied to the example sentence *"The quick brown fox jumps"*

One-Hot Encoding

The bag-of-words model does not specify a numerical encoding scheme for each word and so let us consider how one might represent each word in a sentence or vocabular numerically. Suppose that an algorithm is operating on a language with a vocabulary, $\mathcal{V}$, for example, $\mathcal{V}$ might be the set of all words in the English language. It is assumed that this set is finite, in which case, there are $|\mathcal{V}|$ words which the model must be able to operate on. Let us consider the simplified case that

$$\mathcal{V} = \{\textit{the, brown, fox, quick, fast, dog, large, is}\}$$

and proceed by attempting to represent each word in the vocabulary uniquely.

Suppose that we wished to represent each word with a binary vector, one could use 3-bits to represent each of the words in $\mathcal{V}$, for example, we could encode the vocabulary as follows. Since $|\mathcal{V}| = 8 = 2^3$ this is perfectly possible and, in general, one could represent a vocabulary,

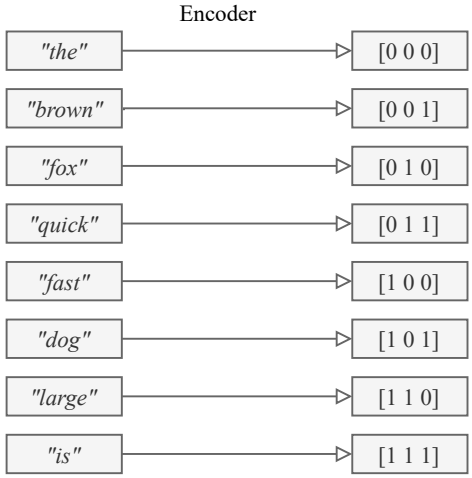

Figure 2.2: An example encoding of the vocabulary $\mathcal{V} = \{\textit{the, brown, fox, quick, fast, dog, large, is}\}$

$\mathcal{V}$, with $\lfloor \log_2(|\mathcal{V}|+1) \rfloor$ bits. An alternative scheme, known as *one-hot encoding*, represents each word using $|\mathcal{V}|$ bits, each vector representation contains a single 1-digit that identifies the word uniquely. Using the example of

$$\mathcal{V} = \{\textit{the, brown, fox, quick, fast, dog, large, is}\}$$

a possible one-hot encoding is shown below.

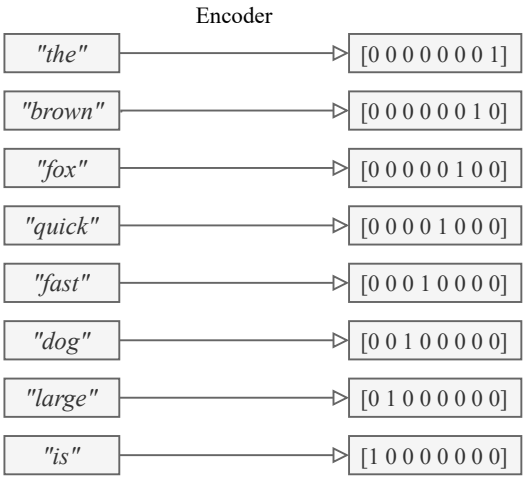

Figure 2.3: An example one-hot encoding of the vocabulary $\mathcal{V} = \{$*the, brown, fox, quick, fast, dog, large, is*$\}$

The virtues of one-hot encoding are made apparent by considering how one might combine a sequence of words into a single vector, a process known as *aggregation*, in which we observe that one-hot encoding, by using orthogonal representations of words avoids ambigious aggregations.

Aggregation

The process of aggregation attempts to summarise a sequence of word representations into a single vector. One could attempt to develop algorithms that operate on the entire sequence directly, however by reducing the dimensionality of the data, learning algorithms will often train more effectively; computations will operate on a smaller amount of data and trends in the data may be easier for the model to observe. In increasing numbers of dimensions, the model will require exponentially more training instances (known as the 'curse of dimensionality' [Bellman, 1966]) to ensure that it can cover a satisfactory portion of the feature space and, hence, aggregation is an important tool in producing effective learning algorithms for human language.

Suppose that we wish the represent the sentence *'brown fox'* using the original coding scheme represented in Figure 2.2 and aggregate it into a single vector that can be supplied to a model. In other words, we wish to devise an aggregation function, f, that operates on a sequence of N, D-dimensional vectors and produces a single D-dimensional vector in return. Many techniques have been devised to do this, but those that disregard the ordering of the vectors are often referred to as 'pooling' techniques of which three are considered here: max pooling, sum pooling and mean pooling. These can be succinctly described as follows.

Max pooling, denoted as $f_{\max}$, simply returns the maximum of the sequence of vectors coordinate-

wise, hence, if $x_{n,d}$ refers to the d-th element in the vector representing the n-th word, then max pooling can be summarised as

$$f_{\max}(\mathbf{x}_1, ..., \mathbf{x}_N) = \begin{bmatrix} \max(x_{1,1}, ..., x_{N,1}) \\ \vdots \\ \max(x_{1,D}, ..., x_{N,D}) \end{bmatrix}$$

Sum pooling, denoted as f_{sum}, takes the sum of the vectors coordinate-wise

$$f_{\text{sum}}(\mathbf{x}_1, ..., \mathbf{x}_N) = \sum_{i=1}^{N} \mathbf{x}_i$$

Mean pooling, denoted as f_{mean}, takes the mean of the vectors coordinate-wise

$$f_{\text{mean}}(\mathbf{x}_1, ..., \mathbf{x}_N) = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i$$

In this case, under the coding scheme in Figure 2.2, max pooling would summarise the phrase *'brown fox'* as

$$f_{\max}\left(\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \right) = \begin{bmatrix} \max(0,0) \\ \max(1,0) \\ \max(0,1) \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \tag{2.1}$$

Sum pooling would give an identical summary

$$f_{\text{sum}}\left(\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \right) = \begin{bmatrix} 0+0 \\ 1+0 \\ 0+1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \tag{2.2}$$

Mean pooling generates a different representation to both max and sum pooling but is a scaled version of the vector $\begin{bmatrix} 0 & 1 & 1 \end{bmatrix}^\top$.

$$f_{\text{mean}}\left(\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \right) = \frac{1}{2} \begin{bmatrix} 0+0 \\ 1+0 \\ 0+1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0.5 \\ 0.5 \end{bmatrix} \tag{2.3}$$

It is critical to notice that the representations in both Equation 2.1 and Equation 2.2 are equivalent to the representation for the word *'quick'* under the scheme in Figure 2.2. In other words, the aggregation has summarised *'brown fox'* into *'quick'* and has, thereby, lost all semantic meaning from the original sentence. This is an example as to why a scheme such as that presented in Figure 2.2 cannot be used with the aggregation techniques introduced.

Conversely, let us consider the results of max, sum and mean-pooling under the one-hot

coding scheme introduced in Figure 2.3 when attempting to aggregate the phrase *'brown fox'*.

$$f_{\max}\left(\begin{bmatrix}0\\0\\0\\0\\0\\0\\1\\0\end{bmatrix}, \begin{bmatrix}0\\0\\0\\0\\0\\1\\0\\0\end{bmatrix}\right) = \begin{bmatrix}0\\0\\0\\0\\0\\1\\1\\0\end{bmatrix} \tag{2.4}$$

Sum pooling would give an identical summary

$$f_{\text{sum}}\left(\begin{bmatrix}0\\0\\0\\0\\0\\0\\1\\0\end{bmatrix}, \begin{bmatrix}0\\0\\0\\0\\0\\1\\0\\0\end{bmatrix}\right) = \begin{bmatrix}0\\0\\0\\0\\0\\1\\1\\0\end{bmatrix} \tag{2.5}$$

Mean pooling generates a different representation to both max and sum pooling but is, again, just a scaled version of the sum-pooled vector.

$$f_{\text{mean}}\left(\begin{bmatrix}0\\0\\0\\0\\0\\0\\1\\0\end{bmatrix}, \begin{bmatrix}0\\0\\0\\0\\0\\1\\0\\0\end{bmatrix}\right) = \begin{bmatrix}0\\0\\0\\0\\0\\0.5\\0.5\\0\end{bmatrix} \tag{2.6}$$

The vectors derived in Equations 2.4, 2.5 and 2.6 are each not present in the one-hot encoding scheme as presented in Figure 2.3 and, as a result, aggregation of the one-hot vectors produces a 'unique' (in the sense that the aggregated vector has no equivalent word in the vocabulary) representation of *'brown fox'*. As a result, the 'optimal' binary encoding scheme, as presented in Figure 2.2, in which words are represented using $\lfloor \log_2(|\mathcal{V}| + 1) \rfloor$ bits is typically not used in NLP and a one-hot representation is favourable in comparison but has its own shortcomings.

The Relative Performance of Different Pooling Techniques At first glance it is unclear as to the relative benefits of each of three pooling techniques introduced; clearly the length of the sentence has a direct impact on the output of the sum pooling. For example, the sum-pooled representation of the sentence *'brown fox fox'* is $\begin{bmatrix}0 & 0 & 0 & 0 & 0 & 2 & 1 & 0\end{bmatrix}^\top$ and the longer the sentence supplied the larger the magnitude of the aggregated vector becomes; this may become problematic since sentences with identical semantic meaning may have very different aggregated vectors depending on their relative length. As an example, *'brown fox brown fox brown fox brown fox'* is sum-pooled as $\begin{bmatrix}0 & 0 & 0 & 0 & 0 & 4 & 4 & 0\end{bmatrix}^\top$ and *'brown fox'* is sum-pooled as $\begin{bmatrix}0 & 0 & 0 & 0 & 0 & 1 & 1 & 0\end{bmatrix}^\top$, yet, arguably, both sentences have more-or-less identical semantic meaning. Mean-pooling attempts to address this flaw by directly accounting for the number of words in the sentence; using the previous example, both *'brown fox brown fox brown fox brown fox'* and *'brown fox'* would be mean-pooled into the vector $\begin{bmatrix}0 & 0 & 0 & 0 & 0 & 0.5 & 0.5 & 0\end{bmatrix}^\top$. In problem domains where the number of words in the sentence is not fixed, such as misinformation-detection, for example, this a clear advantage of mean-pooling over sum-pooling.

Max-pooling has some similar properties to mean-pooling, for example, both *'brown fox brown fox brown fox brown fox'* and *'brown fox'* would be max-pooled into the vector $\begin{bmatrix}0 & 0 & 0 & 0 & 0 & 1 & 1 & 0\end{bmatrix}^\top$, however, in some problem domains there are significant practical differences, specifically, mean-pooling can cause important features to be 'smoothed' out. In other words, by taking the average across all the words in a sentence some important words which may have their meaning masked. In order to illustrate this, consider a model that is tasked with identifying if a sentence mentions the word 'fox' in it. Let us suppose that the model has been trained and is being supplied with a test sentence *'the quick brown fast fox is large'* under max-pooling the resulting aggregated vector would be $\begin{bmatrix}1 & 1 & 0 & 1 & 1 & 1 & 1 & 1\end{bmatrix}^\top$ in which case the model can simply check if there is a one present in the third least-significant bit (which under the example encoding scheme presented in Figure 2.3 corresponds to the word 'fox'). In the case of mean-pooling, the aggregated vector would be $\begin{bmatrix}1/7 & 1/7 & 0 & 1/7 & 1/7 & 1/7 & 1/7 & 1/7\end{bmatrix}$ and, in general, as the length of the sentence increases, under the assumption that the word 'fox' might only occur a single time, the value in the corresponding position in the vector could become negligibly small. In such a problem domain, the model may struggle to correctly classify sentences of increasing length. Therefore, pooling techniques are frequently assessed empirically to determine the most appropriate algorithm for a given dataset and model.

Word Embeddings

The Pitfalls of One-Hot Encoding One-hot encoding produces a distinct binary representation for each word in a vocabulary, $\mathcal{V}$; each encoding is of size $|\mathcal{V}|$-bits and all the entries in the vector are zero except a single entry that identifies the word uniquely. In this sense, one-hot

encoding is 'sparse' since the vast majority of each vector representation is zero. This sparsity is clearly a trade-off, it has the benefit that after aggregating a sequence of vectors a large amount of information is retained however the number of bits needed to represent each word is vast. For example, it is unclear how one might represent the entirety of the English language, for which, $|\mathcal{V}| > 10^5$. Furthermore, each of the vectors are mutually orthogonal and thereby all have a cosine-similarity of zero[3], that is, if the i-th word in the vocabulary is represented as a one-hot vector $\mathbf{x}_i$ then $\forall i, j = \{1, ..., |\mathcal{V}|\}.\mathbf{x}_i \cdot \mathbf{x}_j = 0$. In other words, the representation of each word captures no semantic meaning since synonyms will have the same dot-product value as perfect antonyms, for example, in the one-hot encoding presented in Figure 2.3 no aspect of the encoding captures the fact that *'quick'* and *'fast'* have similar meanings in practice.

The computational burden and lack of semantic understanding inherent to one-hot encoding provided impetus for denser and more language-aware representations; an ideal representation is non-orthogonal in nature and could, thereby, use the representation of the word to capture semantic meaning by ensuring that similar words have representations with a high cosine-similarity.

Fixed Embeddings In order to develop embeddings that capture semantic meaning, one must consider what constitutes similarity between words in the first-place; for example, the word 'dog' in the sentence *'The dog ran away'* could be easily replaced with the word 'cat', but does this mean that the words 'cat' and 'dog' have similar meaning? The difference between having the same meaning and simply being used in the same context is a difficult distinction to make and, in general, requires a high-level understanding of language; should the words 'dog' and 'cat' result in similar embeddings or should they be polar opposites?

Word2Vec [Mikolov et al., 2013] was an attempt at precisely this and learns a lookup-table from words to embeddings that is trained to maximise performance in one of two tasks. One of the formulations of Word2Vec is known as the 'skip-gram' formulation and proceeds as follows: for each sentence in the training corpus, given a word, w, the embedding of w is supplied to a neural network that attempts to predict the words surrounding w in the training instance. For example, if the training instance is *'the dog ran away'* and the word w is 'dog' then the model is tasked with predicting the words surrounding 'dog' i.e. 'the', 'ran' or 'away' with the closest words to 'dog' receiving the most importance. The second formulation, known as the 'continuous bag-of-words' (CBOW), is given a 'bag' of embeddings (i.e. the order is deemed irrelevant) and is tasked with predicting the missing word, for example, the model may be supplied with 'the', 'ran' and 'away' and is expected to predict that 'dog' is missing. Importantly, the authors' suggest that continuous bag-of-words trains more quickly since the model is trained on a bag-of-words rather than on each individual word (as is the case in skip-

[3]The cosine-similarity is a metric of similarity between two vectors, $\mathbf{x}$,$\mathbf{y}$ is $\frac{\mathbf{x} \cdot \mathbf{y}}{||\mathbf{x}||||\mathbf{y}||}$

gram) but, as a result, learns poor embeddings for infrequent words since their importance is typically masked by the words surrounding them in the 'bag'.

GloVe ('Global Vectors for Word Representation') [Pennington et al., 2014] is a different approach to Word2Vec and proceeds instead by computing a co-occurrence matrix, M, of size $|\mathcal{V}| \times |\mathcal{V}|$ for a vocabulary, $\mathcal{V}$, where M_{ij} records the number of times a word x_j has occured within the word x_i's 'local window' (the words within a specified distance of another word); for example, if the window size was one, then the local window of the word 'dog' in *'the dog ran away'* is {'the', 'ran'}. This is an alternative approach to Word2Vec which exclusively operates on local contextual information; GloVe aggregates local statistics across an entire corpus to generate a 'global' perspective on the context of a particular word. The co-occurrence matrix is subsequently used to compute a co-occurrence probability for each pair of words in the vocabulary; the co-occurrence probability P_{ij} of two words x_i and x_j is calculated as $P_{ij} = \frac{M_{ij}}{\sum_k M_{ik}}$. The key conjecture of GloVe is that the ratio of co-occurrence probabilities is deeply informative, for example, consider the ratio of co-occurrence probabilities involving three words, x_i,x_j and x_k,

$$R_{ijk} = \frac{P_{ij}}{P_{jk}}$$

If x_k is closely related to x_i but not to x_j, then R_{ijk} will be very large; whereas if x_k is related to x_j but not x_i then R_{ijk} will be very small; and $R_{ijk} \approx 1$ if x_k is either related to both x_i and x_j or to neither. GloVe proceeds by attempting to learn word embeddings such that the dot product of embeddings is approximately equal to the logarithm of their probability of co-occurrence; since $\log(x/y) \equiv \log x - \log y$ the vector difference between embeddings also encodes the ratio of co-occurrences directly. This has the effect that in the embedding space generated by GloVe, vector differences also encode semantic meaning, for example, if the GloVe encoding is denoted as f, then one might expect relationships of the form $f(\text{'London'}) - f(\text{'England'}) + f(\text{'Egypt'}) \approx f(\text{'Cairo'})$ to be commonplace in the embedding-space.

Word2Vec is a predictive model, whereas GloVe is based on matrix factorisation and, as a result, when used as pre-trained embeddings, may perform differently depending on the problem domain. However, in general, both can be considered as 'fixed' embeddings since, regardless of the context of a word, the same word is always embedded to the same vector. For example, the word 'crane' in *'the crane flew away'* and *'the crane lifts the bricks'* receives the same embedding; and so does the word 'left' in *'my left arm'* and *'I left my book behind'* despite having very different meanings. Words that have different meanings depending on the context but are spelled identically are known as 'homographs' (e.g. 'crane' and 'left' in the examples given) and clearly are a pitfall of both Word2Vec and GloVe. It may be the case that, especially in the pursuit of misinformation-detection, the context of the word is paramount and therefore techniques which do not succumb to the use of homographs are important; an example word embedding model that utilises the context of the word in deriving its embedding is BERT [Devlin et al., 2019].

BERT (Bidirectional Encoder Representations from Transformers) [Devlin et al., 2019] is a language model that is based on the Transformer architecture [Vaswani et al., 2017] and is trained in a 'self-supervised' fashion; in other words, in a similar form to Word2Vec, a large unsupervised dataset, commonly taken from an unlabelled text corpus such as Wikipedia[4], is transformed into a prediction task by masking some of the words in each sentence and training the model to predict the missing word. In the case of BERT specifically, the model attempts to predict the missing word and in doing so learns to represent word embeddings effectively. However, most language models prior to BERT operated as either left-to-right or right-to-left where each word in the sentence is processed to generate a hidden vector which is then utilised by the next word and so on until a final representation for the entire sentence is computed. BERT is, instead, bi-directional and utilises self-attention [Cheng et al., 2016]. Self-attention is a mechanism that relates the importance of different words in each sentence to one another. For example, self-attention may learn to associate a high importance between the words 'crane' and 'it' in the sentence *'the crane flew away but it later came back'* since they refer to the same underlying entity. As a result, BERT may, for instance, return similar embeddings for the words 'crane' and 'it' in such an example, which would not be possible with Word2Vec and GloVe. The embeddings learnt by models like BERT move beyond the simple notions of fixed embeddings and produce contextual embeddings instead; in the context of relatively opaque social media text this may be empowering.

2.1.2 Paraphrasing Models

The task of 'paraphrase generation' has been a longstanding problem in the field of NLP in which an algorithm is tasked with, when supplied a sentence, returning a new sentence with identical meaning but different syntax; for example, a valid paraphrase of the sentence *'Having had breakfast, the dog went for his morning walk'* might be *'The dog set off on his morning stroll after he ate his breakfast'*. As a problem, both paraphrase detection and generation are critical to other longstanding problems in computational linguistics, such as text summarisation and machine translation amongst others. In the context of this thesis, it is of particular interest in evaluating the robustness of misinformation-detection models and in the modelling of potentially malicious social media users that may re-write text in many different ways.

Historically, many approaches have been employed in attempting to solve paraphrase generation. An early approach, as proposed by Barzilay and Lee [Barzilay and Lee, 2003] attempted to align sentences from 'parallel corpora' which, in this case, was a dataset of articles corresponding to the same real-world event, under the assumption that the datasets should contain paraphrased sentences of one another. The Authors attempt to discover re-occurring sentences and underlying patterns; many sentences can be paraphrased by a simple rearrangement of subclauses within the sentence, for example, 'the fox jumped over the fleeing cat' might become 'the fox jumped over the cat which then fled'. A clustering approach is used to match

[4] https://en.wikipedia.org/

sentences from the different articles and, to generate a new paraphase, the learnt patterns of rearrangement are applied to any member of the cluster of the test samples cluster.

Other approaches have drawn inspiration from machine translation, for example, Bannard and Callison-Burch [Bannard and Callison-Burch, 2005] proposed the use of translation datasets for discovering paraphrases. The Authors propose the use of an intermediate language and a double-translation for the generation of paraphrases. For example, suppose the intermediate language is French and the English phrase to be paraphrased is *'the flying cat'*; the model uses an alignment to discover that the translation of the *'the flying cat'* is *'le chat volant'* and then searches for an additional datapoint where *'le chat volant'* is present. If such a datapoint is present, then it performs another alignment to discover the translation of *'le chat volant'* in this datapoint; with enough data, a paraphrased version may exist, for example, a datapoint which translates *'le chat volant'* as the *'the soaring cat'*.

Many modern approaches have attempted to use sequence-to-sequence recurrent neural networks (RNN) [Hochreiter and Schmidhuber, 1997] for the generation of paraphrases; Prakash et al. [Prakash et al., 2016] used an Encoder-Decoder architecture where a long short-term memory (LSTM) unit is used to encode a sequence of words and an LSTM-based decoder is tasked with using the embedded sequence to generate a paraphrase. However, research into attention mechanisms [Cheng et al., 2016] made clear the limitations of RNN-based models; the generation of a single vector representing an entire sequence of words often makes downstream tasks challenging since for long sentences too little information from the start of the sequence is retained. Hence, the application of the transformer architecture, which utilises self-attention, appeared to be a fruitful avenue of research at the time.

T5 Model (text-to-text-transfer-transformer) [Raffel et al., 2020] is a transformer-based model designed to facilitate *transfer learning*[5] in NLP in a direct text-to-text fashion. The T5 model considers every problem that it is applied to as a direct text-to-text problem, this allows the same model to be adapted to tasks like machine translation and text summarisation without modification and it has, as a result, been widely used as a base model before fine-tuning[6] for a variety of tasks. Due to its versatility it has, for example, been used as a model for paraphrase generation and a pre-trained version is used in this thesis.

2.2 Adversarial Machine Learning

Adversarial machine learning is concerned with the manipulation of data inputs in order to deceive a model into an erroneous output, as well as techniques for preventing such errors; the former is typically referred to as an 'attack' and the latter as a 'defense'. Fundamentally,

[5] A paradigm in machine learning that focuses on the development of models that gain re-usable knowledge; for example, BERT learns embeddings that can be re-used for a variety of NLP tasks [Bozinovski and Fulgosi, 1976]

[6] The process of taking a pre-trained model before re-fitting it to a new dataset and adapting the parameters learnt in the pre-training

adversarial machine learning seeks to violate the i.i.d assumption underpinning machine learning by tweaking either the training set or the test set or both. In general, an attack that modifies the training set is referred to as a 'poisoning attack' and is not of primary concern to this thesis, although it may be possible for an adversary to systematically affect the data on social media with the intention of worsening misinformation-detection models, the most likely attack is an individual user that chooses to change their behaviour and thereby the input to the model. An attack of this form, where an adversary tweaks their behaviour to avoid detection, is commonly referred to as an 'evasion attack' since the attacker seeks to escape the detection of the model. There are a plethora of different possible attacks and defenses, the forms considered most relevant to this thesis are discussed.

2.2.1 Attacks

Evasion attacks are typically classified as either 'black-box' or 'white-box'; the former is an attack where the the adversary has no access to the model or any of its parameters and the latter occurs when the adversary has complete access to the model. In the case of a white-box attack, the adversary often uses the gradient of the model's output with respect to its parameters to perturb the input in a highly-specific manner and, thereby, encourage an erroneous output. An example of such a technique is the Fast Gradient Sign Method (FGSM) [Goodfellow et al., 2015] which uses the gradient in the loss-space to find a small perturbation of the input that causes the new-input to cross the classification boundary. Such techniques have been used very successfully, especially in Computer Vision where the perturbation of an image is imperceptible to humans, to fool machine learning models. However, since text is not real-valued when it is passed to the model it is not clear how such a technique which adds noise to the input could be directly applied. Text is discrete in nature and a manipulation in the embedding space can not be easily reversed back through the aggregation and tokenization phases to return to a textual representation. As a result, the primary means of attacking an NLP model is by attacking the text directly rather than any subsequent representation of the text. Although the consequences of a white-box attack are, in general, likely to be more severe and easier to produce, the black-box case is considered of most interest to this thesis; in the context of social media usage, the *de facto* standard adversarial scenario is most probably between a malicious user and a model that they do not have access to.

The application of black-box attacks to NLP models has a further constraint - 'semantic coherence'; the attacked sentence must have more-or-less the same underlying meaning to a human observer since the principle behind an attack, in a misinformation-detection context, is that the attacker conveys the same message to a human recipient whilst causing a change in classification by the detection model.

Character-level attacks An attack where individual characters in a word are manipulated is commonly referred to as a 'character-level' attack and occurs when an adversary attempts to

remove, alter or add a character to a word such that a human observer can instantly recognise the word but such that a model may change its classification decision. An adversary may replace letters with visually similar or even random letters in order to trick the model i.e. 'amazing' might become one of 'åmaƶłɳɠ', 'amazng', 'amaz1ng' or 'amzng' all of which are obvious to a human (even more so with surrounding context i.e. 'that film was amzng') but less so to a model without visual understanding. It has been shown that such attacks can decrease model performance on a variety of NLP tasks by up to 82% [Eger et al., 2019]. Image-based character embeddings [Shimada et al., 2016, Dai and Cai, 2017, Liu et al., 2017] have been used to attempt to bestow models with a visual understanding of the characters on which they operate under the assumption that characters could be embedded such that a large cosine similarity between embeddings implies visual similarity; for example, 'a' and 'å' should have a large similarity and thereby an NLP model may be able to correct character replacements before passing onto a subsequent classification model. Some approaches have used rule-based replacements by attempting to encode common substitutions (for example, i $\rightarrow$ 1 etc.) [Pruthi et al., 2019], such approaches have shown relatively little success since the spectrum of unicode characters is so large and a rule-based approach could not effectively cover all of them.

Word-Level attacks Alternatively, attacks may occur at the word-level, in which a word may be substituted, deleted or additional words appended. If a word is to be substituted it is commonly for a synonym, in which case, such an attack is known as a 'synonym-attack' [Alzantot et al., 2018, Garg and Ramakrishnan, 2020, Li et al., 2020c, Li et al., 2020a, Jin et al., 2019]; the appending of additional words is known as a 'concatenation attack' [Jia and Liang, 2017]. The constraint of semantic-coherence is pertinent when developing techniques for such attacks since the manipulation of a sentence may not, if done without care, retain its original grammar and meaning. The vast majority of papers place a threshold on the number of alterable words in the sentence to attempt to maintain coherence.

Sentence-level attacks Attacks can also be performed at the sentence or phrase level, in which the generalisation ability of the model is tested by an adversary that re-phrases sentences to generate new sentences with identical semantic meaning. It is possible to use a rule-based approach to paraphrasing, for example, one might define a list of rules of the form *'what is'* $\leftrightarrow$ *'what's'*,*'?'* $\leftrightarrow$ *'??'*, *'what'* $\leftrightarrow$ *'which'* etc. Such rules will often preserve semantic coherence but suggested attacks can be filtered out based on a metric of semantic coherence. This is the approach taken by Ribeiro et al. [Ribeiro et al., 2018] in which they define the algorithm `Semantically Equivalent Adversarial Rules` (SEARs) which uses such rules and, having suggested a paraphrase, translates the proposal into a pivot language and then back into the original language; the neural translation model will provide a probability of the translation being correct and if the backtranslated phrase is sufficiently likely then it is taken to be semantically coherent by the SEAR algorithm. Iyyer et al. [Iyyer et al., 2018] also rely on backtranslation as a means of generating data for paraphrasing; the Authors propose an algorithm `Syntatically`

`Controlled Paraphrase Networks` (SCPNs) which is trained on backtranslated sentences and their constituency parses[7], as a result, the resulting SCPN when given a sentence and a desired constituency parse attempts to find a paraphrase conforming to the desired parse [Iyyer et al., 2018]. This, however, relies on the provision of a desired constituency parse as part of the input to the model. The T5 model (see Section 2.1.2) has also been used as a means of generating paraphrases [Nighojkar and Licato, 2021], since the T5 model operates directly on text-to-text tasks, it is well-suited to paraphrase generation as a problem and is considered to be the current state-of-the-art model for this [Li et al., 2020b, Niu et al., 2020, Bird et al., 2020].

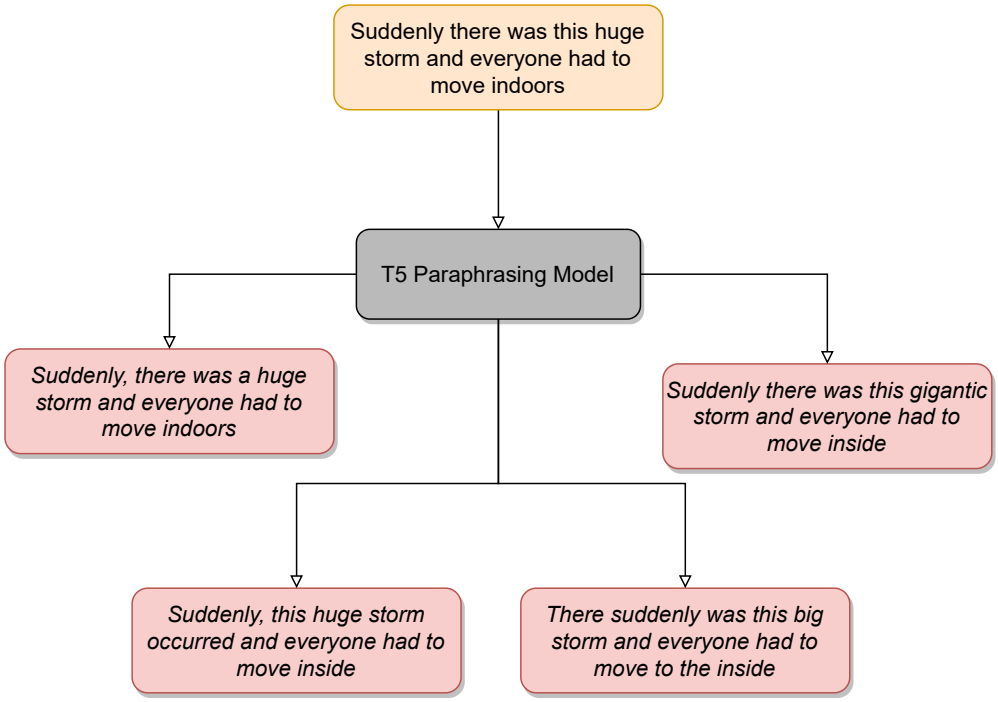

Figure 2.4: Example paraphrases from the T5 model trained on the PAWS dataset [Zhang et al., 2019]; examples are taken from the documentation of the pre-trained model[9]

Genetic attacks Genetic algorithms are an approach to optimisation problems that draw inspiration from the real-world phenomenon of natural selection; candidate solutions are treated as competing members of a species and they 'breed' with one another to produce new candidate solutions. The fittest solutions have the greatest chance of passing on their characteristics and 'mutations' (in which a candidate is randomly perturbed) ensure that a large enough search space is covered by the algorithm. This has been used successfully as means of searching

[7] Constituency parsing is the process of decomposing a sentence into its constituent subphrases an assigning a grammatical category to each word, for example, each word is assigned as being either a noun phrase, verb phrase, etc.

[9] https://github.com/Vamsi995/Paraphrase-Generator

for adversarial cases, for example, GenAttack [Alzantot et al., 2019] is a genetically-inspired black-box attack algorithm that managed to attack state-of-the-art Computer Vision models on the CIFAR-10 [Krizhevsky et al., 2014] and MNIST [LeCun, 1998] datasets with a success rate over 95% and, crucially, using several orders of magnitude fewer queries than the pre-existing most efficient attack algorithm. A high level summary is provided below, in the form of pseudocode, however, it is important to note that GenAttack is specifically formulated for real-valued domains (such as Computer Vision) and its adaptation to discrete problems is an original contribution of this thesis.

Algorithm 1 A high-level overview of the GenAttack algorithm [Alzantot et al., 2019]

Input x ▷ *The data point being attacked*
Input m ▷ *The model being attacked*
Input l ▷ *The targeted label*
generation $\leftarrow$ [ ]
for $i = 1, ...,$Population_Size **do**
▷ *Create the initial generation by adding random noise to* x
    generation[i] $= x +$ RandomNoise()
end for
for $g = 1, ...,$Number_Generations **do**
▷ *Compute the 'fitness' of each member of the gen. and find the fittest*
    $F \leftarrow$ Fitness(generation)
    fittest_member $\leftarrow$ Max(F)
    if m(EliteMember) $= l$ **then** ▷ *A successful attack has been found*
        return EliteMember
    end if
    for $i = 1, ...,$Population_Size **do**
    ▷ *Select parents using the softmax of the fitnesses*
        parent1,parent2 $\sim$ Categorical(Softmax(F))
        ▷ *Breed/'crossover' parents and mutate to give child*
        child = Mutate(Crossover(parent1, parent2))
        ▷ *Add the new child to generation*
        generation[i] = child
    end for
end for

The GenAttack algorithm above is dependent on the following three functions

- `Fitness`: is a metric for the quality of a candidate solution; the Authors decide to reward candidate solutions that increase the probability assigned to the target class, l, and that also decrease the probability assigned to the remaining classes. In this case, the Authors

use

$$\log p(x)_l - \log \sum_{c=0, c \neq l}^{L} p(x)_c$$

where L is the number of classes and $p(x)_j$ is the probability assigned to the j-th class for the input x by the model.

- `Crossover`: a function combining two candidate solutions into a third, ideally better, candidate; the Authors achieve this by using the Softmax[10] of the fitnesses to form a categorical distribution that is sampled from to select each feature at random one-by-one.
- `Mutate`: a function that randomly perturbs a candidate solution; the Authors simply add additional random noise to the candidate to encourage the discovery of potentially successful new candidates.

2.2.2 Defenses

Defending against adversarial attacks, historically, has primarily been through a technique known as a 'adversarial training' [Goodfellow et al., 2015, Madry et al., 2019] in which attacked data points are mixed with the original dataset at training time to encourage the model to generalise to both the attacked and the unattacked examples. Adversarial training is considered to be the most effective, known strategy for defending against an adversary in Computer Vision applications [Pang et al., 2021, Maini et al., 2020, Schott et al., 2018] and has been shown to improve performance of the defended model as well in NLP tasks [Liu et al., 2020, Belohlávek et al., 2018]; however, as of yet, there are no known guarantees as to the extent of this improvement and, furthermore, critically, it relies on the ability of the author of the model to accurately predict the behaviour of the adversary. In other words, if the real-world adversary behaves very differently to the attacked data points then there is not even an empirical understanding as to the model's resulting performance. For example, suppose that adversarial training is used to defend the model against a synonym attack, if the input is perturbed by some other, unknown attack, say a concatenation attack, then the adversarial training will not guard the model against this. Furthermore, there is evidence that adversarial training often reduces performance of the model on the non-attacked data points [Raghunathan et al., 2019, Lamb et al., 2021] by encouraging the model to generalise to too wide of a data distribution.

2.3 Explainable Artificial Intelligence (XAI)

2.3.1 Overview

The proliferation of machine learning has bestowed models with the power to impact the real-world; algorithms are tasked with critical decision-making, including life-or-death healthcare

[10] $\text{Softmax}(\mathbf{x})_i = \frac{\exp(x_i)}{\sum_i \exp(x_i)}$

decisions, and, as a result, research into interpreting such models has become an important subfield within artificial intelligence. The machine learning model cannot afford to be viewed as an opaque 'black-box' when its decisions are to be relied upon. This is as critical to misinformation detection as any other application of machine learning techniques; when a model decides whether or not to publish a piece of information, the authors of the model must be able to interpret its decision and thereby verify its reasoning.

The algorithms proposed in XAI research can be categorised into several classes: *global* or *local* explainers, the former attempts to interpret an entire model and summarise its decision making whereas the latter interprets a single prediction; *intrinsic* or *post-hoc*, the former includes techniques for building explainers as a part of machine learning models whereas the latter studies methods for interpreting a provided model; *agnostic* vs *model-specific*, the former includes techniques that can operate on any 'black-box' model whereas the latter places constraints on the kinds of models that are intepretable (e.g. some techniques may only operate on tree-based models). In the context of this thesis, the explainer must be local, model-agnostic and post-hoc; the model's per-example reasoning must be understandable and a global explainer for misinformation detection is unrealistic since the model is exposed to such a wide array of different topics that a single unified explanation is impossible. Furthermore, a model-specific and/or intrinsic explainer is likely to place needless constraints on the kinds of architectures that are permitted. There are a plethora of techniques developed satisfying these constraints, one of the most popular and widely researched, particularly in NLP, is known as 'Local interpretable model-agnostic explanations' (LIME) [Ribeiro et al., 2016] and is utilised in this thesis.

2.3.2 LIME

The crux of the LIME algorithm is to learn a separate, intrinsically-interpretable model around the data point of interest and use this model, which is only a local approximation to the black-box model, to explain the classification decision. This is achieved by perturbing the data point and observing how the black-box model responds to the perturbed input and then subsequently fitting the local model to this new data. In the case of LIME, this local model is a sparse linear model that attempts to optimise the following

$$\zeta(x) = \underset{g \in G}{\operatorname{argmin}} \quad \mathcal{L}(f, g, \pi_x) + \Omega(g) \tag{2.7}$$

where G is the set of potentially interpretable models (i.e. linear models in this case); $\mathcal{L}(f, g, \pi_x)$ is the loss incurred by the linear approximator, g, in comparison to the black-box model, f, on the region around the example, x, where each perturbed input is weighted by π_x (for example, the Authors define π_x using the radial basis function to ensure that perturbed inputs that are closer to the original example are more important); and $\Omega(g)$ is a measure of the complexity of the local model. This optimisation is referred to, by the Authors, as the 'Fidelity-

Interpretability Trade-off' and attempts to learn a simple but accurate local explanation.

The resulting linear model associates an 'importance' (magnitude of the corresponding weight in the linear model) to each of the features and this constitutes an explanation of the data point; a large negative weight indicates that the feature is negatively associated with the class and the opposite is true of large positive weights. A LIME explanation of an example from the benchmark dataset used in this thesis, the Pheme [Zubiaga et al., 2016] dataset, is shown in Figure 2.5.

Aggregating LIME

The LIME algorithm provides a simple means of explaining the operation of a classifier on a particular data point; yet, how should one, across an entire dataset, attempt to understand the most important words to the model when making a classification decision? A naive approach is to take the mean explained-importance associated with each word in each of the example data points and use this to provide global scores for each of the words observed. Suppose the LIME explained-importance of the j-th word in the dictionary as observed in the i-th example data point is denoted as W_{ij}, then this mean global-explainer is defined as

$$I_j^{\text{Mean-Lime}} = \frac{1}{C_j} \sum_{i=1}^{N} |W_{ij}| \tag{2.8}$$

where C_j is the number of times the word, indexed by j, occurs. Crucially, as observed by the original authors of LIME [Ribeiro et al., 2016], there is difficulty in extending LIME to provide global explanations in such a fashion since the magnitude of weights are not necessarily consistent between instances; that is, a word with a large importance in a particular data point may not retain a large magnitude when transferred to another data point. The approach of the Authors in providing global explanations using LIME, is to, instead, use the square-rooted sum of the feature importances

$$I_j^{\text{LIME}} = \sqrt{\sum_{i=1}^{N} |W_{ij}|}. \tag{2.9}$$

However, work by [van der Linden et al., 2019] has shown that this is unfairly weighted towards words that occur more often and have, as a result, attempted to aggregate LIME explanations across the entire dataset by a different method. The global homogeneity-weighted importance [van der Linden et al., 2019] attempts to rectify the pitfall of both the simple global average (Equation 2.8) and the rooted sum of importances (Equation 2.9) by considering the entropy of a particular word's importances across multiple classes; for example, suppose a word has high importance across multiple classes then its occurence cannot be particularly information-rich since it is not indicative of any one particular class, however, this would still fool both the global average and the rooted sum that would assign very large scores to such a word. Van der Linden et al. proceed by defining the normalized square-rooted importance of the j-th word

with respect to the c-th class

$$p_{cj} = \frac{\sqrt{\sum_{i \in S_c} |W_{ij}|}}{\sum_{c \in L} \sqrt{\sum_{i \in S_c} |W_{ij}|}} \tag{2.10}$$

where S_c denotes the set of all example data points with class c and L is the set of all class labels. The Shannon entropy [Shannon, 1948] of the j-th word is then defined as

$$H_j = -\sum_{c \in L} p_{cj} \log p_{cj} \tag{2.11}$$

and a word with a large Shannon entropy is considered to be 'homogenous', that is, its impact across all the classes is more-or-less the same. Therefore, low entropy words are considered to be more informative and [van der Linden et al., 2019] proceed by weighting the score of each word by a normalised entropy and thereby define the global homogeneity-weighted importance as

$$I_j^H = \left(1 - \frac{H_j - H_{\min}}{H_{\max} - H_{\min}}\right) I_j^{\text{LIME}} \tag{2.12}$$

where $H_{\max}$ and $H_{\min}$ are the maximum and minimum entropies across all the features. This assigns low-importance to words with entropy approaching $H_{\max}$ and large importances as its entropy approaches $H_{\min}$.

2.4 Misinformation Detection

There have been several approaches to solving misinformation detection using machine learning techniques, these can be broadly separated into the following taxonomy: graphical approaches, that utilise the structure of the graph of interactions between users under the assumption that the propagation of misinformation has a distinct spread pattern [Han et al., 2020, Song et al., 2021, Hamid et al., 2020]; purely content-based approaches which only rely on the text itself to inform classification decisions [Zubiaga et al., 2016, Derczynski et al., 2017]; fact-checking approaches which take both a trusted source and the information to be verified as an input before making a decision [Thorne et al., 2018]; and hybrid approaches which utilise a mixture of any of the preceding techniques as well as user-information [Monti et al., 2019]. The graphical approach (also commonly referred to as propagation-based approach) has received much attention [Castillo et al., 2011, Ma et al., 2018, Vosoughi et al., 2018, Wu et al., 2015], yet it has some clear but relatively undiscussed pitfalls. Firstly, it is likely that graphical-based approaches would not be able to scale to social media networks with billions or hundreds of millions of users (as is the case with Facebook and Twitter respectively), the vast majority of datasets used to benchmark these techniques (such as the Twitter15 and Twitter16 datasets [Ma et al., 2017]) contain under 300,000 distinct users. Secondly, machine learning on the propagation structure requires the rumour to have spread in the first place. This would, no doubt, make it much more difficult to stop spreading misinformation; fake news spreads notoriously

quickly at its inception, and, hence, by the time a propagation-based model is in a position to make classification decisions, the rumour may have already dispersed beyond control. Thirdly, there are plenty of sources of information without any propagation information at all, for example, suppose that a developed model is to be used to assess the quality of a news article; it is not obvious how the required propagation information can be obtained if at all. As a result, this thesis chooses to focus on content-based approaches which are deemed to be a much more far-reaching solution.

2.4.1 Dataset

The Pheme dataset [Zubiaga et al., 2016], which is used in this thesis, contains a set of rumours and non-rumours that have been recorded from Twitter and that pertain to nine real-world, breaking news events:

1. The Charlie Hebdo terrorist attack in Paris (January 2015)
2. The rumour that the football player Michael Essien had contracted Ebola (October 2014)
3. The Ferguson, Missouri protests and the shooting of Michael Brown (August 2014)
4. The Germanwings Airplane crash (March 2014)
5. The acceptance of the Gurlitt Art Collection of Nazi loot by the Kunstmuseum in Bern (November 2014)
6. The Ottawa Shooting at Parliament Hill (October 2014)
7. The rumour that the musical artist Prince was to hold a concert in Toronto (November 2014)
8. The rumour that Putin was missing due to the birth of an illegitimate child (March 2015)
9. The siege of a Sydney cafe (December 2015)

Each of these topics contains a number of tweets that have each been assigned as being either true, false or unverified and is thereby a three-way classification problem, such a task is often referred to as 'veracity classification' since the aim of the model is discern if the given information is truthful. An example is shown in Figure 2.5 alongside its LIME explanation and Table 2.1 shows the breakdown of the three classes for each topic.

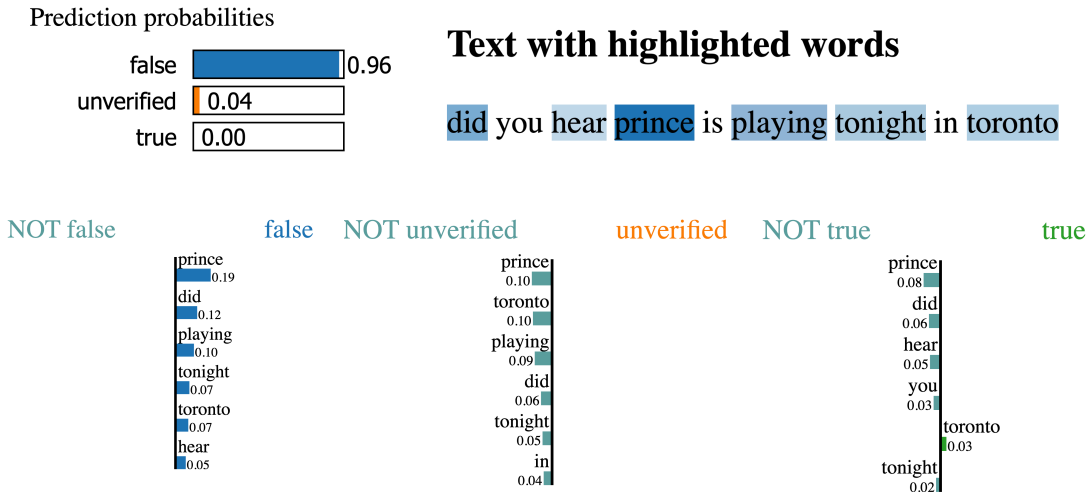

Figure 2.5: An example LIME explanation from a misinformation-detection model applied to the sentence *'did you hear prince is playing tonight in toronto'* which pertains to a false rumour spread on Twitter in November 2014

Count	Veracity		
Topic	**False**	**Unverified**	**True**
Charlie Hebdo	116	149	193
Ebola Essien	14	0	0
Ferguson	8	266	10
Germanwings Crash	111	33	94
Gurlitt	0	2	59
Ottawa Shooting	72	69	329
Prince Toronto	222	7	0
Putin Missing	9	117	0
Sydney Siege	86	54	382

Table 2.1: The number of occurrences of each class with respect to each event in the dataset, the order of events matches the listing of topics

2.4.2 Previous approaches for the Pheme dataset

The Pheme dataset was also presented at the International Workshop on Semantic Evaluation[11] (SemEval) in 2017; five research groups contributed submissions which were evaluated against a hidden test set. The approaches of each of the five teams are briefly summarised with their respective results shown in Table 2.2; each of the teams are referred to by an acronym.

[11] https://semeval.github.io/

The IKM team [Chen et al., 2017] used a convolutional neural network [LeCun et al., 1989] to learn their own set of word embeddings (as opposed to using a set of pre-trained word embeddings before fine-tuning) which is max-pooled to aggregate the word embeddings of each word in the sentence; this is then passed to a multi-layer perceptron which makes the classification decision.

The NileTMRG [Enayet and El-Beltagy, 2017] used a set of manually engineered features, such as the number of followers that the user has or the number of URLs in the tweet, and feed this to a support vector machine [Cortes and Vapnik, 1995] which is then used to learn a classification boundary.

The DFKI team [Srivastava et al., 2017] similarly to [Enayet and El-Beltagy, 2017] used a set of manually engineered features which are used with one of three classifiers that they trained: a maximum entropy classifier [Malouf, 2002]; a naive Bayes model [Frank and Bouckaert, 2006]; or a Winnow classifier [Littlestone, 1987].

The ECNU team [Wang et al., 2017] combined manually engineered features with metadata regarding both the tweet and the user; this is then concatenated with word embeddings of the tweet (GloVe [Pennington et al., 2014] and Word2Vec [Mikolov et al., 2013] are both experimented with) and finally fed to a classifier. The Authors experiment with logistic regression, support vector machines [Cortes and Vapnik, 1995], random forests [Breiman, 2001] and AdaBoost [Freund and Schapire, 1997].

The IITP team [Singh et al., 2017] similarly to several of the other approaches, combine manually engineered features and experimented with three classifiers: naive Bayes, support vector machines and decision trees.

Team	**Root Mean Squared Error (RMSE)**
DFKI DKT	0.845
ECNU	0.736
IITP	0.807
IKM	0.763
NileTMRG	0.672

Table 2.2: The results of each of the five teams in the SemEval-2017 competition on the Pheme dataset [Derczynski et al., 2017]

Crucially, none of these approaches surpassed the most-common-class baseline in which the same class (the most frequently occurring) is predicted for all data points in the test set; this is, as observed by the Authors, indicative of the difficulty of training a model for this dataset. The Authors have not released the test set used and, as a result, direct comparison is not possible.

Furthermore, the Authors attempt to test the submitted models on withheld topics, this is not a realistic use case since no reasonable model can discern truth on unobserved topics and would thereby be merely making uneducated guesses based on patterns-of-text. The Authors also use the root mean-squared error as the main metric of choice, in this thesis, accuracy is considered to be more appropriate for a three-way classification and therefore, the Table 2.2 is included merely for context and a different evaluation strategy is pursued.

Chapter 3

Methods

The problem of misinformation-detection, in accordance with the Pheme dataset [Zubiaga et al., 2016], is formulated as a three-way classification task. A model, f, when supplied with an item of text, x, is tasked with returning a label from the set $\{0, 1, 2\}$ whereby the label, 0, for example, identifies the text as false, 1, corresponds to an 'unverified'[1] item of text and, 2, identifies an item of text as being true. However, as is reviewed in Section 2.1, most models cannot operate directly on text and thereby must process the text in some way such that it is amenable to inference. As a result, this Chapter begins by outlining a concrete, end-to-end pipeline that transforms text into a numerical format and subsequently into a classification decision. Importantly, this pipeline decomposes the task into a sequence of independent stages, each of which can be accomplished by different algorithms and, thereby, acts as a pre-cursor to the experimentation that follows in the subsequent Chapter which analyses the impact of the algorithmic choices at each stage. Secondly, this Chapter defines, mathematically, the adversarial domain that is assumed; this concrete framework is subsequently utilised in defining two separate adversarial algorithms: a 'fixed' and a genetic algorithm (known as the `Misinformer`), each of which attempt to manipulate the input to each benchmark model such that it becomes convinced of a particular classification decision. In order to develop the genetic algorithm, which relies on understanding the classification decisions of a so-called 'surrogate' model, the Global Homogeneity-Weighted Importance [van der Linden et al., 2019] (as introduced in Section 2.3.2) is extended to encode per-class information and thereby allow the adversary to target the adversary in a particular fashion. Instead of simply compelling the adversary into a misclassification one may wish to change the attacked model's classification decision to a particular class; specifically, in the fourth Chapter, both adversarial attacks are used to attempt to convince the attacked models that a false or unverified statement is in fact true.

[1] The text cannot be identified with certainty as being either true or false

3.1 Model Design

Concretely one may view an overall misinformation-detection model as a composition of the following four stages:

1. **Tokenization**: a tokenizer is supplied with a body of text and decomposes each sentence into a set of words or subwords, referred to as 'tokens', for example, the sentence *'the quick brown fox'* can be split into *['the', 'quick', 'brown', 'fox']*, each of these are assigned an integer, known as an 'ID', representing the word. The tokens along with the dictionary containing the IDs can then be passed onto an embedder which transforms these tokens into a set of a continuous vectors. In this project, only two different tokenizers are considered, BERT typically utilises a WordPiece tokenizer that splits words in sub-words and, as result, when using BERT embeddings, the WordPiece tokenizer from the huggingface library[2] is used. However, both Word2Vec and GloVE utilise a standard word tokenizer in which a sentence is simply split into each constituent word and, hence, when using either of these embeddings a word tokenizer is used.

2. **Embedding**: having been supplied with a set of tokens, the embedder is tasked with returning a meaningful continuous representation of each token. For example, suppose an embedder is supplied with a sentence comprising of L tokens and this particular embedding function, f_e, embeds tokens into a vector space $\mathbb{R}^D$ then the function f_e, is of the form $f_e : \mathbb{N}^L \to \mathbb{R}^{D \times L}$; in other words, the embedder returns a tensor of size $D \times L$. In this thesis, the three embedding models introduced in Section 2.1: BERT, Word2Vec and Glove are considered.

3. **Aggregation**: however, since the length of each sentence L is variable and $D \times L$ can be large, it can be difficult to easily operate on a variety of sentence lengths and inference can be become more challenging as a result. Hence it is commonplace to utilise an aggregator function, f_a, that is tasked with summarising a tensor of size $D \times L$ into a single vector of size D, in other words, f_a is of the form $f_a : \mathbb{R}^{D \times L} \to \mathbb{R}^D$ and attempts to distill the entire sentence into a single embedding. As was introduced in Section 2.1, both mean- and max-pooling are utilised in this thesis, as well as both Convolutional Neural Networks [LeCun et al., 1989] (CNNs) and Recurrent Neural Networks [Hochreiter and Schmidhuber, 1997] (RNNs).

4. **Classification**: finally, a model is tasked with using a single vector embedding of the input text to arrive at a classification decision; the previous three stages are specific to an NLP-based problem, however, this final stage can utilise any classification techniques that operate on continuous-valued data. In this thesis, however, only two classifiers are investigated: a logistic regression model and neural networks.

[2] https://huggingface.co/

The overall pipeline and the algorithms considered at each stage are summarised in Figure 3.1.

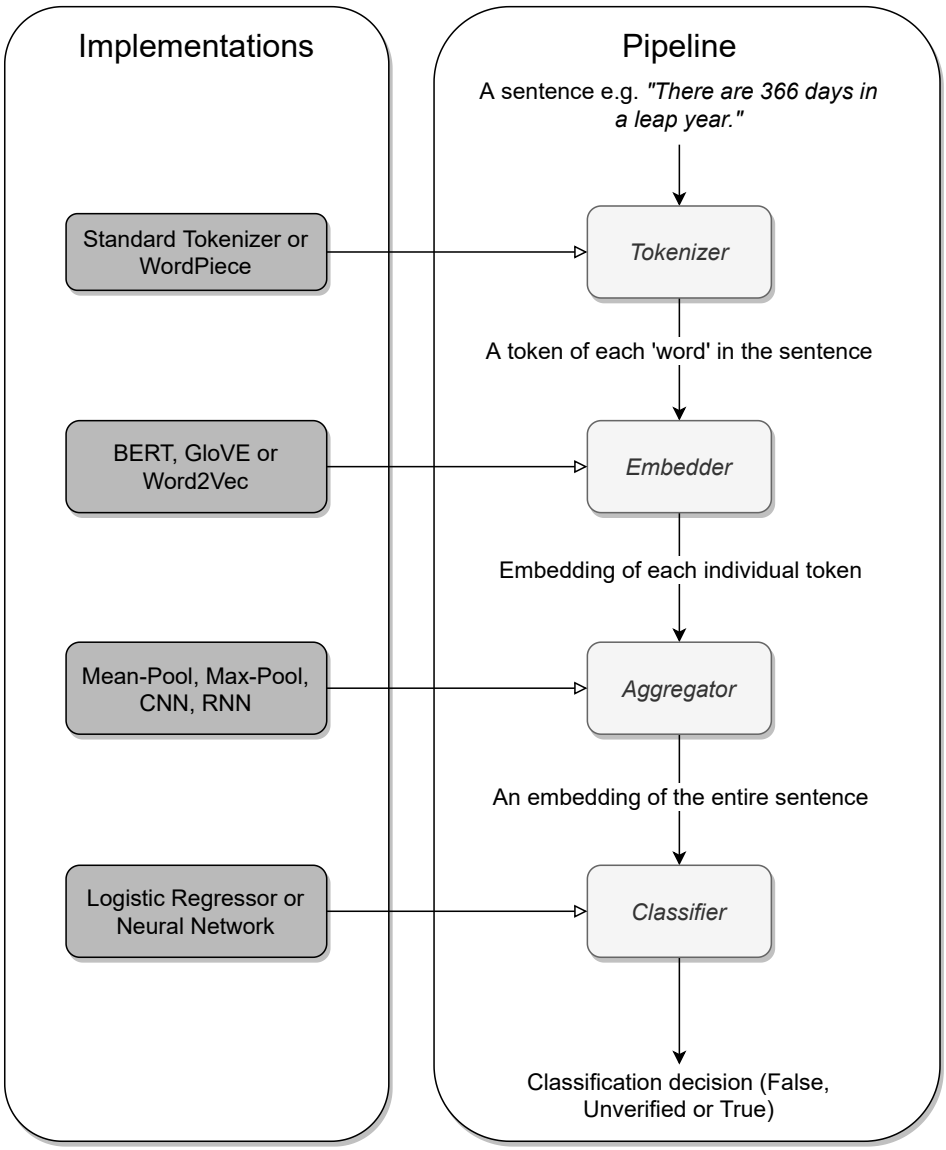

Figure 3.1: A diagram illustrating the entire pipeline from processing a piece of text to the classification decision; implementations used at each of the stages of the pipeline are listed on the left.

3.2 Adversary

3.2.1 Adversarial Domain Definition

The social-media context is relatively unique to adversarial machine learning and allows for the development of novel algorithms; let us begin by considering the information a well-equipped adversary may have access to. Firstly, one may consider social-media to be a 'fully-observable' domain, in some sense, since when a user contributes to the social network the result is, in general, accessible to all parties. For example, if a user has a public account, then upon posting a piece of text anybody on the network may view it. Suppose that a social network decides to release a misinformation-detection model and, in order to avoid malicious users manipulating the model, the company decides not to directly display the classifcation decision of the model to the user, instead it simply hides their post from the network if it is deemed to be a lie. However, due to the network being fully-observable the user can, simply, from the perspective of another user observe their own account and immediately discern the decision of the model. Furthermore, this can be readily automated through the use of bots or automated agents on the social network. Hence, one must assume that the adversary is operating, at the least, in a black-box, since they do not have direct access to the model, and hard-label[3] scenario.

There are further ramifications of the fully-observable property of social-networks; a well-equipped adversary has, in fact, access to the data upon which, presumably, the model they are attacking has been trained on[4] and can, thereby, manually label the data to arrive at a subset, if not the entirety, of the training data used by the model. This may be of particular use to an adversary attempting to attack a model with respect to a recent, breaking topic on the social network since the data at the time may be relatively small and, thereby, easy to label.

Finally, it is assumed that the adversary is targeting a particular decision from the model. The adversary could, for example, attempt to manipulate the model into misclassifying a piece of true news as being false, however, for the purposes of this thesis the most damaging case is considered to be the reverse, where a falsehood is manipulated into convincing the model that it is true. Hence, it is assumed that the adversary to the misinformation-detection model has access to the entirety of the training data and is operating within a black-box, hard-label framework and is attempting to target a 'true' classification-decision from the attacked model.

In more precise terms, an adversary, f_{adv}, perturbs a set of unseen data points $\{(x'_1, y'_1), ..., (x'_T, y'_T)\}$ that are fed to the attacked model, f_m; both models have access to the same set of labelled training examples $\{(x_1, y_1), ..., (x_N, y_N)\}$. In the untargeted adversarial case, the adversary

[3] Hard-label refers to the adversarial scenario where the adversary has access to the decision of the attacked model but not the probabilities or confidences with which the decision was made

[4] Under the assumption that the model is trained on data from the network upon which it is to act

f_{adv} seeks to simply maximise the following objective

$$\max \sum_{i=1}^{T} \mathbb{1}[f_m(f_{\text{adv}}(x_i')) \neq y_i'] \tag{3.1}$$

However, the targeted adversarial case is more realistic in the misinformation-detection context since a malicious adversary is likely to be aiming for a particular classification outcome, hence, for a target label, l, the objective is redefined as

$$\max \sum_{i=1}^{T} \mathbb{1}[f_m(f_{\text{adv}}(x_i')) = l] \tag{3.2}$$

In our case, the label l will correspond to the 'true' classification since this is the most damaging class of targeted attack in which misinformation is allowed to propagate without detection. Clearly, however, there must be restrictions placed on f_{adv}, since, under this notation, a valid adversary may simply proceed as follows

$$f_{\text{adv}}(x') = x_i \quad \text{s.t.} \quad y_i = l \tag{3.3}$$

In other words, the adversary could when supplied with any data point to perturb simply return an example from the training set with the desired label, under the assumption that the attacked model has learnt to correctly classify it. For example, it would clearly be non-sensical for an adversary tasked with deceiving a model that *'the earth is flat'* is a true statement to return *'there are 365 days in a non-leap year'*. This would be in violation of the 'semantic coherence' principle outlined in Section 2.2.1 and thereby there must be constraints on valid adversaries. In an abstract sense, the constraint is that x and $f_{\text{adv}}(x)$ must encode identical meanings to a human observer, in Computer Vision attacks this is typically ensured by controlling the amount of perturbation the adversary can legally apply to the input. Although the notion of semantic coherence is difficult to formally verify, the adversarial algorithms introduced, by analogy to Computer Vision, proceed by limiting perturbation of the input to ensure that coherence is maintained, a process referred to as 'perturbation control'. However, in order to develop algorithms for targeted attacks, let us begin by extending the Global Homogeneity-Weighted Importance [van der Linden et al., 2019] introduced in Section 2.3.2 to interpret the impact of language on each class and subsequently use this to encourage a particular classification decision.

3.2.2 Extending Global Homogeneity-Weighted Importance

The Global Homogeneity-Weighted Importance [van der Linden et al., 2019] can be used to aggregate the LIME importance of each word in each example sentence across an entire dataset. The defining equations are recalled for convenience below, however for a full explanation see

Section 2.3.2. The LIME score of a word, indexed by j in the i-th data point is denoted as W_{ij} and the score of the j-th word is computed as

$$I_j^{\text{LIME}} = \sqrt{\sum_{i=1}^{N} |W_{ij}|} \tag{3.4}$$

This score is normalised to compute a pseudo-probability of the j-th word belonging to the c-th class

$$p_{cj} = \frac{\sqrt{\sum_{i \in S_c} |W_{ij}|}}{\sum_{c \in L} \sqrt{\sum_{i \in S_c} |W_{ij}|}} \tag{3.5}$$

and this pseudo-probabilty is used to calculate the Shannon entropy

$$H_j = -\sum_{c \in L} p_{cj} \log p_{cj} \tag{3.6}$$

A low entropy indicates that the word considered is important to a specific class and is used to calculate the Global Homogeneity-Weighted Importance as

$$I_j^H = \left(1 - \frac{H_j - H_{\min}}{H_{\max} - H_{\min}}\right) I_j^{\text{LIME}} \tag{3.7}$$

Crucially, however, this results in a single value per-word that attempts to aggregate its 'global' importance, that is, how impactful it is is causing any classification decision; this is useful certainly, however, to an adversary attempting to cause a particular classification decision it is not especially useful. For example, if this scoring system were to be used, then an adversary may be aware that the words *'breaking'* and *'BBC'* may be important but it will be completely unaware as to the manner in which it is important. An adversary could, for example, probe all the important words until the desired classification decision is arrived at, however, this would increase the number of queries of the attacked model and thereby create a less effective adversary. As a result, let us extend this system to encode per-class information.

Firstly, the LIME importance W_{ij} encodes important information in the sign of its value; $W_{ij} < 0$ implies that in the i-th data point, the j-th word contributed negatively to the true class y_i. However, Equation 3.4 discards this information by taking the absolute value. As a result, let us begin by redefining Equation 3.4:

$$I_{cj}^{\text{LIME-TARGETED}} = \sqrt{\max\left(\sum_{i \in S_c}^{N} W_{ij}, 0\right)} \tag{3.8}$$

where S_c denotes the set of data-points with labelled with the c-th class. It is critical to retain the square-root (see Section 2.3.2) since [van der Linden et al., 2019] suggest that it is less biased than the mean towards frequently-occurring words, however, the maximum with zero

must be taken to avoid a negative square-root. The adversary is only concerned with positive-associations and hence, this scheme, assigns a score of zero to words which are negatively associated with the desired class. The equation for p_{cj} is retained and the per-class score is computed as

$$I_{cj}^{\text{H-POS-TARGETED}} = \left(1 - \frac{H_j - H_{\min}}{H_{\max} - H_{\min}}\right) I_j^{\text{LIME-TARGETED}} \tag{3.9}$$

As a result, a large value is indicative of a positive association across the example data points between the j-th word and the c-th class. Furthermore, negative associations can be isolated as well by calculating the following score instead

$$I_{cj}^{\text{LIME-NEG-TARGETED}} = \sqrt{\max\left(-\sum_{i \in S_c}^{N} W_{ij}, 0\right)} \tag{3.10}$$

and, then use this in Equation 3.9 to compute a negative association instead, denoted as i $I_{cj}^{\text{H-NEG-TARGETED}}$. This novel extension is subsequently denoted as the `Targeted-Importance Score` (or 'T-score') and both the negative and positive associations are utilised in the attack algorithms, these are referred to as the negative and positive T-scores respectively.

3.2.3 Constituent attacks

The adversarial attacks developed are 'multi-level' attacks, that is, they combine several constituent attacks: a concatenation attack, a paraphrasing attack and a character-level attack to perturb the input; each of these attacks are formulated as randomised algorithms that can, when supplied with an input, provide a different perturbation on subsequent calls.

Concatenation attack The positive T-score encodes valuable information as to which words might be concatenated onto the sentence to alter the classification decision of the attacked model towards a particular target. For example, for a target class, l, a large value of $I_{lj}^{\text{H-POS-TARGETED}}$, indicates that the j-th word is positively associated with the l-th clas and thereby concatenating the word j may alter the classification decision without considerably impacting semantic coherence. The softmax[5] of the T-scores of the observed words with respect to the targeted class provides a probability distribution across the set of words.

[5] $\text{Softmax}(\mathbf{x})_i = \frac{\exp(x_i)}{\sum_i \exp(x_i)}$

Algorithm 2 A randomised algorithm for performing concatenation attacks that utilises Targeted-Importance scores

Param. l ▷ *The target class*
Param. C_1 ▷ *The number of words to concatenate*
Param. D ▷ *The dictionary of J words*
function ConcatenationAttack(x) ▷ *x is the input string*
  $\mathbf{p} \leftarrow \text{Softmax}(\left[I_{l1}^{\text{H-POS-TARGETED}} \quad \dots \quad I_{lJ}^{\text{H-POS-TARGETED}} \right])$
  for $i = 1...C_1$ **do**
    $j \leftarrow$ sample from Categorical($\mathbf{p}$)
    $x \leftarrow \text{Concatenate}(x, D[j])$
  end for
  return x
end function

Paraphrasing attack The paraphrasing attack is simply the result of querying a pre-trained T5 model [Raffel et al., 2020, Nighojkar and Licato, 2021] supplied by the huggingface library [6] that has been trained on the 'Paraphrase Adversaries from Word Scrambling' (PAWS) dataset [Zhang et al., 2019]. The T5 model when supplied a query statement returns a number of paraphrased versions of the sentence; the number of sentences returned is a parameter to be selected.

Character-level attack The character-level attack works by attempting to perturb the words most negatively associated with the target class. For example, suppose the T-scores recognise that the word 'flat' in *'the earth is flat'* is negatively associated with a true-classification (i.e. its prescence is indicative that the content is unlikely to be true), the algorithm could utilise this to perturb 'flat' and return, say, *'the earth is f1at'* and attempt to fool the model as a result. Similarly to the concatenation attack, the softmax function is used to generate a categorical distribution, however, in the case of the character-level attack the distribution is over the number of words in the sentence rather than the number of words in the dictionary. This is used to select the word to be manipulated and then one of three operations is randomly performed: a deletion, insertion or substitution, in the case of the latter two, a random letter is either inserted or substituted respectively.

[6] https://huggingface.co/

Algorithm 3 A randomised algorithm for performing character-level attacks that utilises Targeted-Importance scores

Param. l ▷*The target class*

Param. m ▷*The maximum num. edits to an attacked word (the Levenshtein distance)*

Param. C_2 ▷*The maximum number of words to attack*

Param. D ▷*The dictionary of J words*

$\mathbf{p} \leftarrow \text{Softmax}\left(\left[I^{\text{H-NEG-TARGETED}}_{lI_{w_1}} \quad \dots \quad I^{\text{H-NEG-TARGETED}}_{lI_{w_S}}\right]\right)$

function CHARACTERATTACK(x) ▷*x is the input string with words $w_1, w_2, ..., w_S$ and IDs $I_{w_1}, ..., I_{w_S}$*

  for $i = 1...C_2$ **do**

    $i \leftarrow$ sample from Categorical($\mathbf{p}$) ▷*Sample index of the attacked word*

    p sample from Uniform$(0, 1)$

    if $p < 1/3$ **then**

      w_i has a character randomly deleted

    else if $p < 2/3$ **then**

      w_i has a character randomly substituted

    else

      w_i has a character randomly inserted

    end if

    Replace w_i with the attacked version in x

  end for

  return x

end function

3.2.4 Fixed Adversary

The three previously defined attacks can be combined into a single attack that returns a set of perturbed inputs as follows:

Algorithm 4 A function that, given a single input string, returns a set of perturbed inputs utilising the paraphrase attack, concatenation and character-level attacks

Param. P ▷*The number of paraphrases to include in the set of perturbed sentences*
Param. N ▷*The number of perturbed sentences to return*
function GENERATEATTACK(x) ▷*x is the input string*
  $X' \leftarrow [\ \ ]$
  for $i = 1...P$ **do**
    Append `Paraphrase`(x) to X'
  end for
  for $i = 1...(N - P)$ **do**
    Append x to X'
  end for
  for $i = 1...N$ **do**
    $X'[i] \leftarrow$ `ConcatenationAttack`(`CharacterAttack`($X'[i]$))
  end for
  return X'
end function

The algorithm above returns a set of M strings that have been perturbed from the original sentence, x. A fixed adversary[7] can be defined by simply querying the attacked model, f_m, on each of the M perturbed strings; if a single sentence causes the targeted classification, then it is considered a failure on behalf of the attacked model, f_m.

Algorithm 5 An algorithm for the fixed adversary

Param. l ▷*Target class*
Param. f_m ▷*The attacked model*
for $i = 1...T$ **do**
  if $f_m(x'_i) \neq l$ **then**
    $G \leftarrow$ `GenerateAttack`(x'_i) ▷*Generate a batch of perturbed sentences*
    for a in G **do**
      if $f_m(a) = l$ **then**
        return a ▷*Attack has succeeded*
      end if
    end for
  end if
end for

[7] An adversary that does not adapt to the model it is attacking

3.2.5 Genetic Adversary

The genetic algorithm, GenAttack [Alzantot et al., 2019], was introduced in Section 2.2.1 as a genetically-inspired adversarial algorithm, however, crucially it is not applicable to the problem of adversarial misinformation-detection for several reasons.

Firstly, GenAttack is formulated for tasks operating on continuous data (such as Computer Vision based tasks), whereas misinformation-detection is operating on text which is fundamentally discrete. GenAttack generates adversarial examples by adding uniform real-valued noise to the input, clearly, this is not possible for text since this would require adding noise to the embedding thereby violating semantic coherence; furthermore, such a scheme is not easily reversible either and so the resulting adversarial text could not be isolated.

Secondly, GenAttack assumes a soft-label setting in which the adversary has access to the probability assigned to each class by the model; this is not a reasonable scenario to assume in the social-media context since the provider of the platform is unlikely to knowingly provide an adversary with such valuable information.

Therefore, `Misinformer` proceeds by reformulating GenAttack to rectify both of these issues. The algorithm is reformulated for the discrete case, by redefining the procedures for population generation, breeding and mutation in GenAttack; and by using a surrogate-model[8], f_s to avoid the assumption of a soft-label scenario. In the following algorithm, assume that f_s returns a vector of probabilities, i.e. $f_s(\textit{`the earth is flat'}) = \begin{bmatrix} 0.01 & 0.02 & 0.97 \end{bmatrix}$, for example, indicates that the surrogate model has assigned a probability of $0.01, 0.02$ and 0.97 to each of the three classes respectively.

[8] A model trained by the adversary on the data they have access to from the network; this is separate from the model it is attacking

Algorithm 6 `Misinformer` - a genetic algorithm extended from GenAttack [Alzantot et al., 2019] to the discrete, hard-label case

Param. l ▷ *Target class*

function MISINFORMER(x) ▷ *The data point to be attacked*

if $f_m(x'_i) \neq l$ **then** ▷ *Only proceed if the model doesn't already predict l*

$G_1 \leftarrow$ `GenerateAttack`(x)

▷ *Generate first gen. using the multi-level attack* G_1^n *is the n-th member*

for gen $= 1, ..., \text{MAX\_GENERATIONS}$ **do**

$p_{jn} \leftarrow f_s(G_1^n)_j$

▷ *Use surrogate-model to compute likelihood of each class across the generation*

▷ p_{jn} *is the prob. of class* j *under the surrogate for* n*-th memb. of* G

$f_n \leftarrow \log p_{ln} - \sum_{j \neq l} p_{jn}$

▷ *Compute the fitness of each member of the population*

$k \leftarrow \underset{n \in \{1,...,N\}}{\text{argmax}} \ f_n$

▷ *Find the fittest member*

if then$f_m(g_k) = l$

▷ *Check if the fittest member is a successful attack, if so return*

return g_k

end if

$p_{\text{selection}} = \text{Softmax}(f_1, ..., f_N)$

▷ *Use softmax to compute a prob. of selecting each member as a parent*

for $j = 1, ..., N$ **do**

$\text{par}_1, \text{par}_2 \leftarrow \text{Categorical}(p_{\text{selection}})$

▷ *Sample two parents from the corresponding categorical distribution*

$\text{child} = \texttt{Breed}(\text{par}_1, \text{par}_2)$

▷ *Breed the two parents to produce a child*

$\text{child} = \texttt{Mutate}(\text{child})$

▷ *Mutate the child*

$G^j_{\text{gen}} = \text{child}$

▷ *Add the child to the new generation*

end for

end for

end if

end function

The above algorithm is relatively unadapted from GenAttack [Alzantot et al., 2019], other than the use of the surrogate rather than the attacked model for computing the fitness of each member of the population. The definitions of the functions `Breed` and `Mutate` must be adapted from GenAttack, however, since, for example, the Authors use a mutation operation that adds

random noise to the child and is, thereby not applicable. Furthermore, it is, as of yet, unclear as to how this scheme will ensure semantic coherence; the perturbations applied must be carefully controlled in the `Breed` and `Mutate` functions.

The `Breed` function There are a number of practical concerns that the `Breed` function must carefully address to ensure coherence. Firstly, the paraphrasing attack may result in two parents that are of very different syntactic forms. The Authors of GenAttack simply use the Softmax of the fitness of the two parents to select each feature at random; let us consider how this would fail for a text-based problem. Suppose that the first parent is *'there are 365 days in a non-leap year'* and the second is *'in a non-leap year, there are 365 days'*, the GenAttack approach could conceivably return the following child, for example, *'in are non days in a are leap days'*[9] which, even if it causes the desired misclassification, is entirely non-sensical. As a result, one must carefully consider the case that either of the parents have been paraphrased.

Secondly, it is the purpose of the `Breed` function to ensure that the most effective attacks are passed onto the child and therefore, if the sentences are of the same syntactic form, the algorithm must utilise this to favour the fitter of the parents. If both parents have not been paraphrased then `Breed` proceeds by analogy to GenAttack and selects each word with probability derived from the Softmax of the fitnesses. The more effective parent will, most likely, contribute a greater proportion of its words to the child, however, the sampling of each word ensures that there is room for variation with each call of the function.

Algorithm 7 The `Breed` function used in `Misinformer`

function BREED(par_1, par_2) ▷ *The two parents being bred*
  $\mathbf{p} \leftarrow \text{Softmax}(\text{fitness}(\text{par}_1), \text{fitness}(\text{par}_2))$
  Child $\leftarrow$ ''
  if par_1 or par_2 is paraphrased **then**
    Child $\leftarrow$ Choose($[\text{par}_1, \text{par}_2]$, prob. $= \mathbf{p}$)
    ▷ *Select one of the parents using the Softmax of the fitnesses*
  else
    for $i = 1, ..., \text{Length}(\text{par}_1)$ **do** ▷*By definition Length(par_1) = Length(par_2) in this case*
      Child[i] $=\leftarrow$ Choose($[\text{par}_1[i], \text{par}_2[i]]$, prob. $= \mathbf{p}$)
      ▷ *Select child's i-th word from one of the parents using the Softmax of the fitnesses*
    end for
  end if
  return Child
end function

The `Mutate` function Similarly to the `Breed` function, the `Mutate` function must ensure that adversarial inputs are not perturbed without restriction. Suppose that the following pre-

[9] Assume that tokenization splits 'non-leap' into 'non' and 'leap'

mutation child has been bred *'the qu1ck brown fox'*; when *'quick'* was perturbed into *'qu1ck'* it was subject to the constraints of the character-level attack (e.g. the number of modifications was restricted). This is crucial since it ensures that the attacks don't produce unrealistic perturbations, however, a naive approach may mutate *'the qu1ck brown fox'* by simply reapplying the original attacks and, for example, the child could become *'the q1k brown fox'*[10]. Furthermore, in the case of the concatenation attack, this would allow sentences to progressively increase in length as more and more words are appended. Since the concatenation attack maintains coherence under the assumption that a human observer can discern out-of-place words, the coherence of the attacked and original sentence would quickly disappear with progressive generations, under the naive scheme, as the number of original words vanish in proportion. The proposed `Mutate` function attempts to mutate the supplied sentence at the word-level and ensures coherence as follows, each case is exemplified using the example *'the qu1ck brown fox spaghetti'* (assume that 'spaghetti' has been concatenated and that 'qu1ck' has been character-attacked)

- If a word is to be mutated and it is has been previously unattacked, it is attacked with a new character attack, however, an existing character-attacked word must be reset to its original. For example, if 'brown' is to be attacked then 'qu1ck' is reset to 'quick' e.g. *'the qu1ck brown fox spaghetti'* → *'the quick brwn fox spaghetti'*
- If the word selected has previously been character-attacked, then it is reset to its original and has a new character-attack applied; e.g. *'the qu1ck brown fox spaghetti'* → *'the quick brown fox spaghetti'* → *'the quic brown fox spaghetti'*
- If the word selected was itself a concatenation onto the original sentence, then it is to be removed and replaced with a new concatenation e.g *'the qu1ck brown fox spaghetti'* → *'the qu1ck brown fox'* → *'the qu1ck brown fox lasagne'*

Hence, the following algorithm for `Mutate` is derived.

[10] If, for example, the character attack selects a previously attacked word

Algorithm 8 The `Mutate` function used in `Misinformer`

function MUTATE(c) ▷*The child to be mutated*
original ▷*The original unnattacked version of c or the paraphrased version if c has been paraphrased*
char_inds ▷*The indices of the character-attacked words*

for $n = 1, ..., \text{NUM\_MUTATIONS}$ **do**
index ← RandomInteger(0, Length(c))
▷ *Select a random word in c to mutate*
if c[index] is unattacked **then** ▷*If it is neither a concatenation or character-attacked word*
reset_index ← Choose(char_inds)
c[index] ← `CharacterAttack`(c[index])
c[reset_index] ← original[reset_index]
▷ *Perform a fresh character attack and reset a previously character-attacked word*
else if index ∈ char_inds **then** ▷*If the word has been previously character-attacked*
c[index]← `CharacterAttack`(original[index])
▷ *Replace the word with a character attack of its original*
else if c[index] is a concatenated word **then**
c[index] ← `ConcatenationAttack`()
▷ *Replace the word with a newly selected concatenation-attack word*
end if
end for
return c
end function

Chapter 4

Experiments

This chapter begins by evaluating the model pipeline designed in Section 3.1 with a variety of embeddings, aggregation algorithms and classifiers used; the Pheme dataset [Zubiaga et al., 2016] acts as the main benchmark for this. However, the adversarial case is the primary focus of this thesis, and, hence, supervised performance is only discussed briefly. Subsequently, the trained models are used to calculate the distribution of Targeted-Importance scores across the vocabulary; it is shown that a small subset of the vocabulary account for the largest scores and that, as a result, there is reason to believe only a small subset of words are used in making classification decisions. If this is the case, it would be indicative of the potential susceptibility of these models to adversarial attacks. The derived Targeted-Importance scores are used with both the fixed adversary (Section 3.2.4) and genetic adversary (known as the `Misinformer` algorithm) in attempting to attack the best-performing trained models; it is shown that the `Misinformer` algorithm, even with extremely restricted perturbations, can reduce the absolute accuracy of the classifier by $\sim$22% and with less restricted perturbations can reduce the best-performing model to predicting the target class for all test examples. Finally, it is shown that the `Misinformer` algorithm is resistant to adversarial training even when the attacked model is trained on examples conforming to the exact type of adversarial attack.

4.1 Performance on the Pheme Dataset

The models developed were each trained using five different random seeds and early stopping where if performance on a separate validation set, as determined by the accuracy and cross-entropy loss, ceased to improve or decreased then training was terminated. The primary objective of the data collected is to answer the following preliminary research questions:

- *Are there any systematic advantages provided by specific embeddings, aggregation algorithms or classifiers? If so, which of these contributes the most to performance and why?*

- *Which misclassifications are the most common? Are there any classes which are easily confused by the majority of models?*

The five best performing algorithms, as seen in Table 4.1, by accuracy on the test set, were all trained using BERT as their embedding. This is the clearest pattern amongst the model's trained, which, otherwise, seem to show relatively little performance difference between different aggregation algorithms and classifiers; although many of the models showed increased performance under logistic regression rather than an MLP (e.g. 'GloVe+RNN+LogReg' scored 0.705 *vs.* 'GloVe+RNN+MLP' with 0.635) this is likely due to unstable training rather than systematic improvements since the reverse pattern is sometimes also observed. Overall, the data may be a indicative that BERT's contextualised embeddings are advantageous in misinformation-detection.

The six best performing models over the test set have their normalised confusion matrices plotted in Figure 4.1 and the findings can be broadly summarised as follows: for the majority of the models (five of six), they performed worse at correctly predicting true statements than at correctly identifying false statements; this is primarily because many of the models struggled with the distinction between unverified and true information. Many of the unverified statements in the dataset are mostly unverified due to the time of posting rather than the content itself, for example, *'Terrible news... Airbus A320 from Barcelona to Dusseldorf has crashed in France (Germanwings Flight 9525).'*, is taken from the dataset and contains perfectly true information, but at the time of posting, was not verifiable. This is a difficult nuance for a model to discern and the distinction between the classes in such a case is rarely obvious; later statements with similar content, such as *'BREAKING - A Germanwings Airbus A320 plane reportedly crashed in the region of Digne (French Alps) #Flightradar24 - French TV'*, are classified as true and this contributes to the models' difficulties in distinguishing between true and unverified data points.

Architecture			Accuracy	
Embedding	**Aggregation**	**Classifier**	**Train**	**Test**
BERT	MeanPool	MLP	0.928 ± 0.027	0.740 ± 0.014
BERT	MeanPool	LogReg	0.798 ± 0.006	0.745 ± 0.027
BERT	MaxPool	MLP	0.903 ± 0.025	0.719 ± 0.010
BERT	MaxPool	LogReg	0.857 ± 0.009	0.739 ± 0.020
BERT	CNN	MLP	0.985 ± 0.033	0.682 ± 0.075
BERT	CNN	LogReg	0.972 ± 0.019	0.671 ± 0.094
BERT	RNN	MLP	0.908 ± 0.143	0.696 ± 0.149
BERT	RNN	LogReg	0.908 ± 0.043	0.758 ± 0.025
GloVe	MeanPool	MLP	0.796 ± 0.102	0.673 ± 0.056
GloVe	MeanPool	LogReg	0.681 ± 0.007	0.663 ± 0.031
GloVe	MaxPool	MLP	0.683 ± 0.153	0.625 ± 0.124
GloVe	MaxPool	LogReg	0.720 ± 0.007	0.688 ± 0.025
GloVe	CNN	MLP	0.978 ± 0.011	0.534 ± 0.056
GloVe	CNN	LogReg	0.956 ± 0.036	0.595 ± 0.058
GloVe	RNN	MLP	0.884 ± 0.037	0.635 ± 0.069
GloVe	RNN	LogReg	0.848 ± 0.034	0.705 ± 0.030
Word2Vec	MeanPool	MLP	0.785 ± 0.077	0.698 ± 0.082
Word2Vec	MeanPool	LogReg	0.455 ± 0.007	0.445 ± 0.018
Word2Vec	MaxPool	MLP	0.792 ± 0.066	0.683 ± 0.041
Word2Vec	MaxPool	LogReg	0.693 ± 0.006	0.654 ± 0.028
Word2Vec	CNN	MLP	0.948 ± 0.084	0.601 ± 0.061
Word2Vec	CNN	LogReg	0.979 ± 0.010	0.642 ± 0.026
Word2Vec	RNN	MLP	0.645 ± 0.199	0.549 ± 0.107
Word2Vec	RNN	LogReg	0.759 ± 0.095	0.656 ± 0.042
MC-Baseline			0.437 ± 0.014	0.438 ± 0.017

Table 4.1: The performance of each combination of embedding, aggregation algorithm and classifier. MLP denotes a 'multi-layer perceptron' and the train and test accuracy are taken over five different random seeds with the mean and standard deviation indicated. The 'MC-Baseline' is the classifier that simply returns the most commonly occuring class in the training set for all predictions.

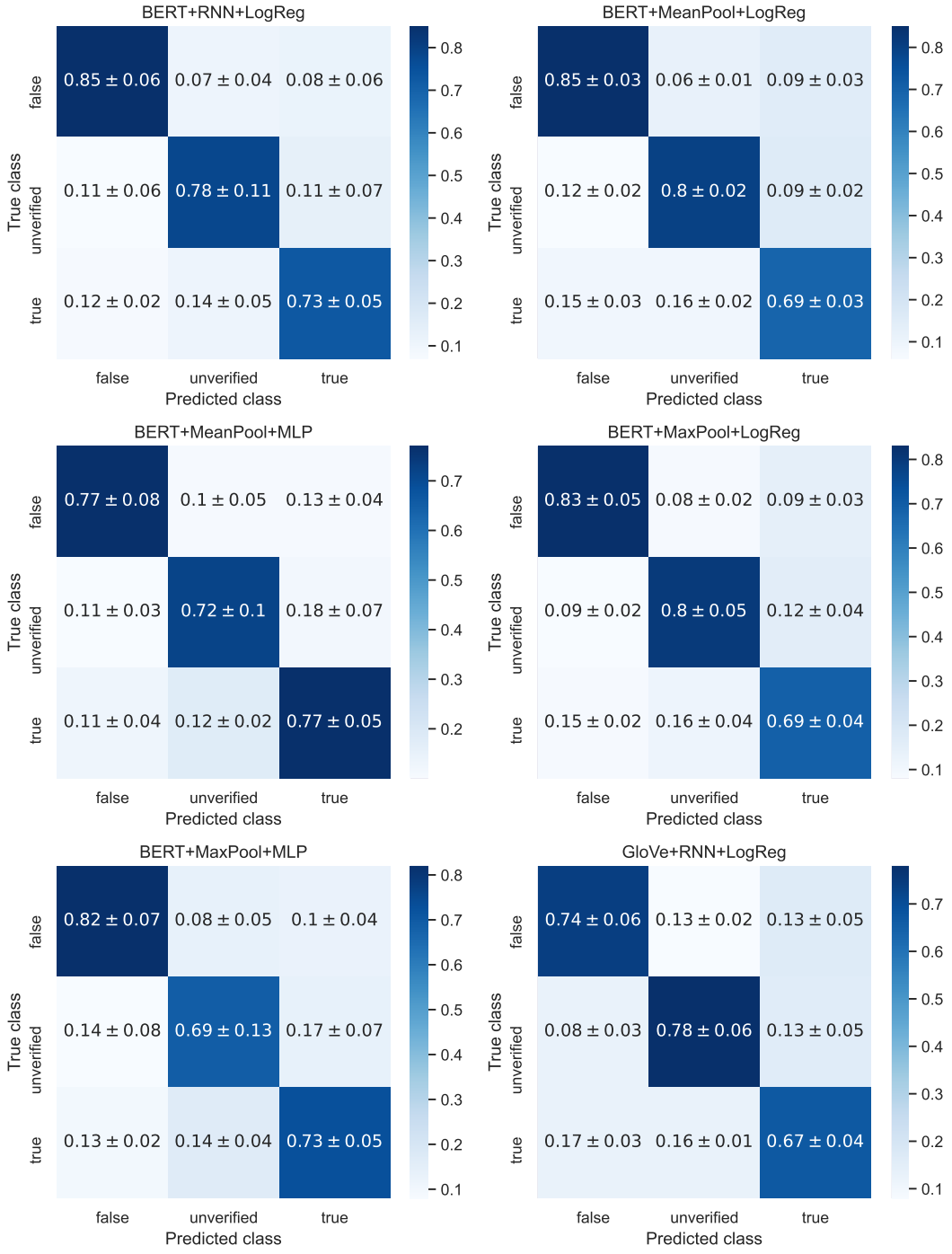

Figure 4.1: The normalised confusion matrices of the six best performing models on the test set.

4.2 Model Intepretation

The distribution of Targeted-Importance Scores (see Section 3.2.2) is elucidated by the plots in Figure 4.2 which exemplify that the distribution of T-Scores has a particularly long-tail. 97.9% of the 3044 words observed at train-time were assigned a T-Score for the true class of less than one, yet the scores range all the way to a maximum of just under four. This is an indication that a small number of words have a disproportionately large impact on the decision making of the models.

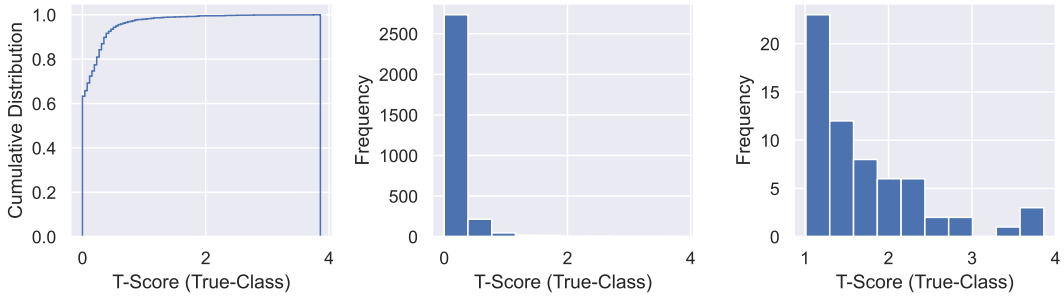

Figure 4.2: Left-to-right: a cumulative histogram showing the distribution of T-Scores for the true class; a histogram across the entire distribution of T-Scores for the true class; a histogram for those words with a T-Score greater than 1.

The ten words with the largest Targeted-Importance scores for the true class are plotted in Figure 4.3; the true class is particularly pertinent since these words could be used to convince a model to classify a given piece of text as true. Many of the words are specific to some of the events in the dataset, particularly those for which a majority of the data-points are true; for example, 'Paris' and 'Sydney' are in reference to the Charlie Hebdo attack and Sydney Cafe siege respectively which, as shown in Table 2.1, contain mostly true tweets. This is further indication that the models developed may be relying on the presence of individual words in making classification decisions.

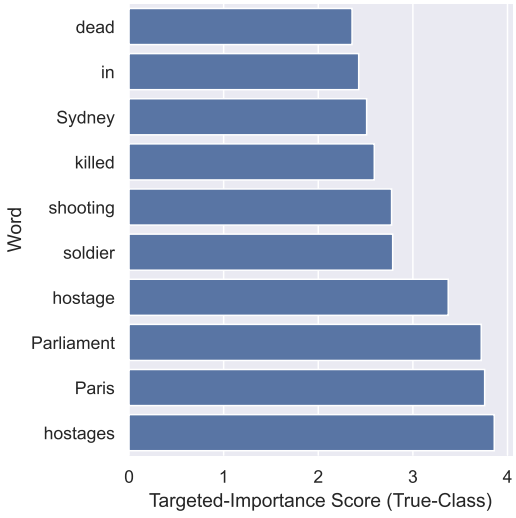

Figure 4.3: The ten words with the largest Targeted-Importance scores for the true class

4.3 Adversarial Performance

4.3.1 Attack

The two adversarial attacks introduced, the fixed adversary and the `Misinformer` algorithm, attempt to manipulate a trained model into changing its classification decision to a particular class. The fixed adversary achieves this by generating a batch of adversarial examples, each of which are passed to the model and a successful attack occurs when one of the batch causes a change in classification, whereas `Misinformer` uses a genetic algorithm to search for an adversarial example and therefore uses a variable number of queries on each attack.

Parameters The two algorithms combine character-level attacks, paraphrasing attacks and concatenation attacks and thereby share a set of common parameters which manipulate the behaviour of the underlying attacks, namely:

- A boolean flag that controls if the paraphrasing attack is to be used (denoted as **'Para.?'** in subsequent tables).
- The number of concatenations that the concatenation attack is allowed to append to the perturbed sentence (subsequently denoted as **'Concats'**).
- The maximum number of edits to an attacked word that the character-level attack is allowed to make.

Metrics The fixed adversary is principally evaluated for its impact on the attacked model. The primary metric for this is the 'attack success rate' (ASR) which is the percentage of attacks that succeed. That is, for an attacked model, f_m, target label, l, adversary f_a and set $X_{\neg l} = \{x. \quad f_m(x) \neq l\}$ of data points which are not classified as l by the model, the ASR is defined as

$$\text{ASR} = \frac{1}{|X_{\neg l}|} \sum_{x \in X_{\neg l}} \mathbb{1}[f_m(f_a(x)) = l]. \tag{4.1}$$

The genetic adversary is also evaluated on its efficiency; a more capable adversary is able to attack the model with fewer queries. In this application, the maximum number of queries the adversary can make to the attacked model is fixed and for different settings of the parameters the number of queries used is recorded and evaluated.

Fixed adversary Across the parameters attempted, the fixed adversary successfully attacked between 38% and 60.5% of test examples it was exposed to (see Table 4.2); even with the most restricted form of attack, in which no paraphrasing or concatenations are used, and only a single edit is allowed for each character-attacked word, the adversary still manages to succesfully attack 41% of examples and reduce the accuracy on the test set from ∼73% to ∼65%. In the experimentation used, the fixed adversary generated batches of 32 candidate adversarial

examples, and evaluated the attacked model on each of these. The accuracy is significantly reduced across all the parameter examples, with the most impactful attacks reducing accuracy to ∼58%.

Genetic adversary The results in Table 4.3 pertain to the attack of the genetic adversary, `Misinformer`, on the best-performing model trained in this work, 'BERT+RNN+LogReg' and indicates how even the best-performing model is prone to adversarial examples. Using the most restrictive parameter-setting, with no paraphrasing, concatenations and only a single edit in the character attack, `Misinformer` succeeds on ∼73% of test examples and reduces accuracy to ∼53% with a mean of 1.7 and median of 2 queries per successful attack. The less-restrictive settings frequently succeeded on >90% of test examples and resulted in a test accuracy that was extremely close to the minimum possible accuracy (the frequency of the target class). In other words, the genetic adversary manages to extend the fixed adversary in such a manner that accuracy is greatly reduced with far fewer evaluations and results in an attacked model that approximates a simple baseline. A set of qualitative examples of such attacks is included in Figure 4.4.

Impact of Perturbation Control Parameters Across both the fixed and genetic adversaries, increasing numbers of concatenations had the most pronounced impact on the strength of the attack, for example, with other parameters accounted for, the difference in ASR between zero and four concatentations is ∼20% (under the genetic adversary), whereas, for comparison, the difference between an edit-distance of one and three is negligible across most of the measurements. However, an edit distance of one, without any other attacks, is still a reasonable attack, particularly in the genetic case, when it reports an ASR of 73%. The efficacy of such a simple attack that can only edit single characters, and, in particular, can only edit a single character across three words in the sentence, is noteworthy. Moreover, paraphrasing seems to have little effect on the efficacy of the adversary, this is likely due to, in the current formulation of `Misinformer`, the paraphrasing attack is only used in the spawning of the initial generation; incorporating paraphrasing into the `Mutate` function may restore some efficacy.

Attack Parameters			Test Accuracy		
Para.?	**Concats.**	**Edit Dist.**	**Attacked**	**Min.**	**ASR**
False	0	1	0.653	0.437	0.412
False	0	2	0.644	0.437	0.423
False	1	1	0.626	0.437	0.477
False	1	2	0.626	0.437	0.489
False	2	1	0.607	0.437	0.549
False	2	2	0.608	0.437	0.535
False	3	1	0.582	0.437	0.604
False	3	2	0.586	0.437	0.605
True	0	1	0.653	0.437	0.405
True	0	2	0.660	0.437	0.382
True	1	1	0.629	0.437	0.474
True	1	2	0.638	0.437	0.452
True	2	1	0.606	0.437	0.537
True	2	2	0.614	0.437	0.520
True	3	1	0.599	0.437	0.558
True	3	2	0.600	0.437	0.568

Table 4.2: The results of applying the fixed adversary with various parameters to the three best-performing models using the surrogate model 'BERT+MeanPool+LogReg'. The attack success rate (ASR) is listed; the number of queries on all attacks and on exclusively successful attacks (mean and standard deviation are indicated) are listed; the minimum possible accuracy, which is the frequency of the target class, is also listed.

Attack Parameters			Test Accuracy			Num. Queries		
Para.?	**Concats.**	**Edit Dist.**	**Attacked**	**Min.**	**ASR**	**Mean All Attacks**	**Mean Succ. Attacks**	**Median Succ. Attacks**
False	0	1	0.530	0.437	0.733	4.649 ± 9.674	1.708 ± 4.609	2
False	0	2	0.526	0.437	0.743	4.328 ± 9.288	1.483 ± 3.830	2
False	0	3	0.532	0.437	0.733	4.449 ± 9.501	1.485 ± 4.028	2
False	1	1	0.516	0.437	0.781	4.195 ± 9.146	1.791 ± 4.861	2
False	1	2	0.505	0.437	0.802	3.628 ± 8.389	1.430 ± 3.669	2
False	1	3	0.507	0.437	0.802	3.667 ± 8.523	1.473 ± 4.011	2
False	2	1	0.491	0.437	0.850	2.923 ± 7.493	1.249 ± 3.390	1
False	2	2	0.484	0.437	0.866	3.048 ± 7.582	1.570 ± 4.317	1
False	2	3	0.491	0.437	0.850	3.079 ± 7.692	1.415 ± 3.907	1
False	3	1	0.489	0.437	0.850	2.983 ± 7.515	1.313 ± 3.473	1
False	3	2	0.478	0.437	0.888	2.597 ± 6.853	1.346 ± 3.642	1
False	3	3	0.482	0.437	0.877	2.634 ± 6.966	1.260 ± 3.387	1
False	4	1	0.468	0.437	0.920	2.054 ± 5.786	1.155 ± 2.933	1
False	4	2	0.464	0.437	0.930	1.979 ± 5.669	1.201 ± 3.258	1
False	4	3	0.459	0.437	0.936	2.087 ± 5.838	1.373 ± 3.809	1
True	0	1	0.534	0.437	0.727	4.730 ± 9.875	1.733 ± 4.935	2
True	0	2	0.530	0.437	0.743	4.333 ± 9.486	1.487 ± 4.340	1
True	0	3	0.534	0.437	0.722	4.726 ± 9.718	1.662 ± 4.368	2
True	1	1	0.516	0.437	0.781	4.052 ± 8.918	1.634 ± 4.284	2
True	1	2	0.522	0.437	0.749	4.274 ± 9.248	1.488 ± 3.919	2
True	1	3	0.516	0.437	0.786	3.877 ± 8.801	1.508 ± 4.120	2
True	2	1	0.503	0.437	0.797	3.900 ± 8.849	1.661 ± 4.644	1
True	2	2	0.511	0.437	0.802	3.574 ± 8.455	1.372 ± 3.794	1
True	2	3	0.493	0.437	0.845	3.642 ± 8.362	1.951 ± 5.193	1
True	3	1	0.491	0.437	0.845	3.368 ± 8.045	1.659 ± 4.522	2
True	3	2	0.486	0.437	0.872	2.942 ± 7.192	1.521 ± 3.738	1
True	3	3	0.482	0.437	0.872	2.983 ± 7.562	1.565 ± 4.455	1
True	4	1	0.474	0.437	0.893	2.493 ± 6.718	1.299 ± 3.583	1
True	4	2	0.478	0.437	0.888	2.526 ± 6.725	1.272 ± 3.356	1
True	4	3	0.474	0.437	0.893	2.638 ± 6.985	1.451 ± 4.126	1

Table 4.3: The results of applying `Misinformer` with various parameters to the best-performing model 'BERT+RNN+LogReg' using the surrogate model 'BERT+MeanPool+LogReg'. The attack success rate (ASR) is listed; the number of queries on all attacks and on exclusively successful attacks (mean, standard deviation and median are indicated) are listed; the minimum possible accuracy, which is the frequency of the target class, is also listed.

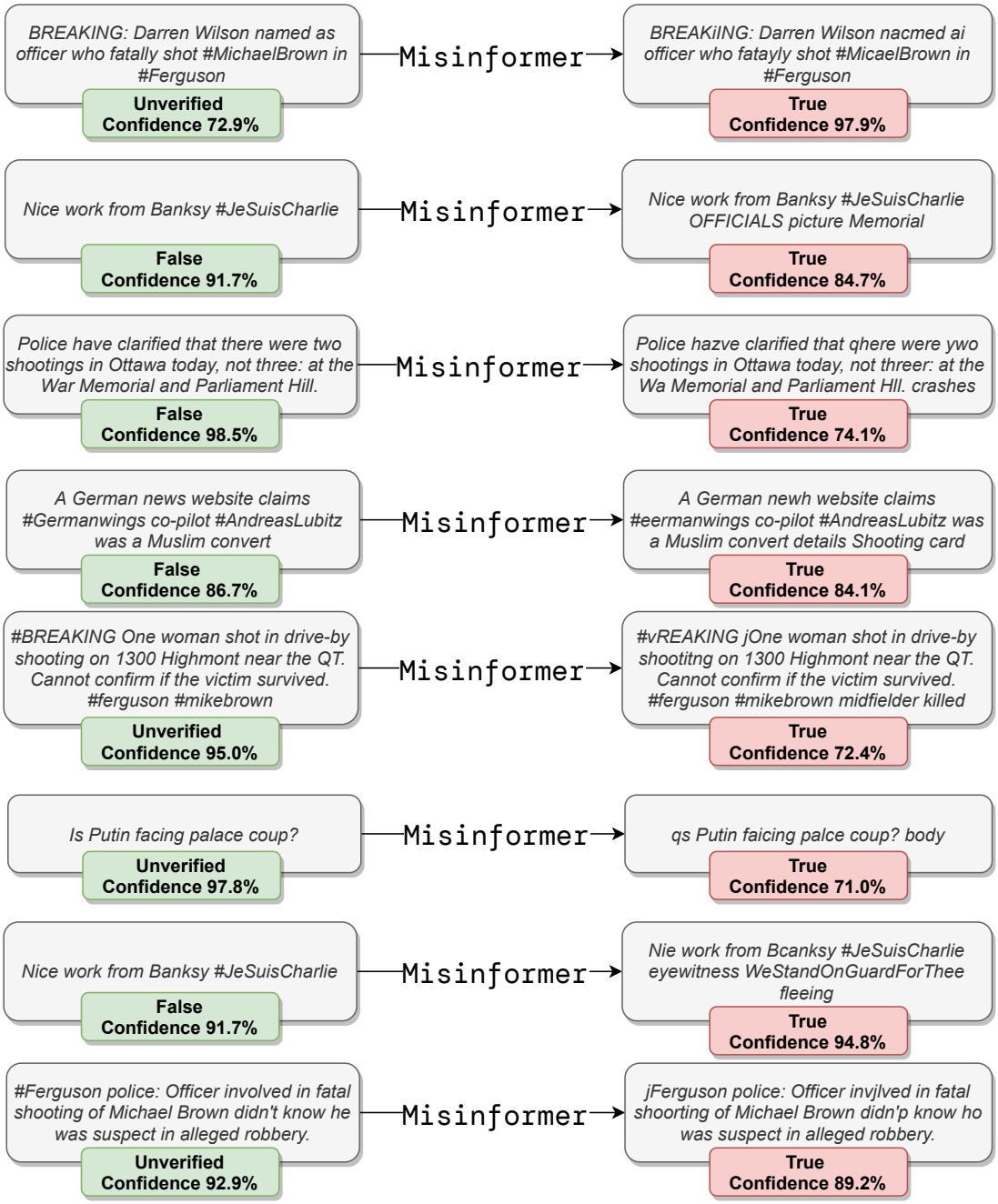

Figure 4.4: A set of example successful adversarial attacks produced by the `Misinformer` algorithm; the original tweet is indicated on the left hand side alongside its true class and the confidence of the model; the right hand size shows the attacked text and the corresponding confidence of the model.

4.3.2 Defense

The primary means of defending a model that is subjected to an adversarial attack is by adversarial training, where example attacks are mixed with the original dataset to encourage the model to generalise to both and, thereby, improve its performance on the attacked examples. The eight best performing model-architectures on the original dataset were retrained using adversarial training with examples from the precise attack they would be subjected to and the efficacy of each attack was recorded; the results are shown in Figure 4.4.

The accuracy on the unattacked data is more-or-less the same with and without adversarial training, in both cases the test accuracy was ∼70% for the majority of models. However, adversarial training does not decrease the susceptibility of the model to the adversarial attack; the ASR under the adversarially trained models remains high, with the most-restrictive attacks recording an ASR of >75% and the least restrictive recording an ASR of >88% with the attacked accuracy decreasing accordingly.

The main effect of adversarial training is a momentary defense where the number of queries required for an attack to succeed increases but the defense does not withhold. Across all the attack parameter settings, the mean and median numbers of queries for successful attacks increased under adversarial training. Assuming a null hypothesis that the number of queries for a successful attack is the same under standard and adversarial training, then the p-values under a two-sided T-test are 4.481×10^{-7} and 0.0332 for the mean and median respectively.

Attack Parameters			**Test Accuracy**				**Num. Queries**		
Para.?	**Concats.**	**Edit Dist.**	**Unattacked**	**Attacked**	**Min.**	**ASR**	**Mean All Attacks**	**Mean Succ. Attacks**	**Median Succ. Attacks**
False	0	1	0.691 ± 0.037	0.521 ± 0.082	0.437	0.754 ± 0.192	6.908 ± 4.084	3.371 ± 1.058	3.143 ± 1.215
False	0	2	0.726 ± 0.030	0.514 ± 0.064	0.437	0.787 ± 0.103	5.599 ± 2.429	2.630 ± 1.656	2.750 ± 1.035
False	1	1	0.713 ± 0.030	0.486 ± 0.045	0.437	0.858 ± 0.085	4.343 ± 1.438	2.530 ± 1.309	2.312 ± 1.100
False	1	2	0.720 ± 0.034	0.495 ± 0.050	0.437	0.852 ± 0.105	3.992 ± 1.949	2.045 ± 0.697	1.750 ± 0.707
False	2	1	0.726 ± 0.036	0.490 ± 0.053	0.437	0.857 ± 0.095	4.493 ± 2.380	2.400 ± 1.703	2.000 ± 1.528
False	2	2	0.717 ± 0.041	0.489 ± 0.051	0.437	0.867 ± 0.103	3.938 ± 1.871	2.121 ± 1.204	2.000 ± 1.000
False	3	1	0.714 ± 0.021	0.487 ± 0.050	0.437	0.878 ± 0.096	4.285 ± 2.477	2.367 ± 1.079	2.000 ± 1.069
False	3	2	0.714 ± 0.029	0.488 ± 0.051	0.437	0.885 ± 0.106	3.390 ± 2.584	1.698 ± 0.850	1.500 ± 0.756

Table 4.4: The aggregated results of training the eight best-performing models with adversarial training and attacking each using the genetic adversary with various parameters; no paraphrasing is used in the attacks listed.

In conclusion, `Misinformer` is slightly less efficient against adversarially-trained models but retains its efficacy. An adversary may require more attempts to succeed in its attack but will do so in a reasonable number of attempts nonetheless.

Chapter 5

Conclusion

5.1 Summary

The performance of a model on a carefully-curated dataset may not be a fair representation of its real-world efficacy. Social media, and the internet at large, can be a particularly adversarial domain where users are constantly adapting their behaviour to evade automated censorship; users will, for example, send coded forms of hate speech which are recognisable to humans but difficult to systematically detect with a model or algorithm. As misinformation-detection models are yet to be released into the public domain, this thesis has sought to understand what pitfalls, if any, models must overcome in order to attain an acceptable level of robustness to the ever-changing landscape of user-behaviour.

In order to investigate this, two novel contributions were made: that of the Targeted-Importance Score (see Section 3.2.2) which is a novel metric for globally aggregating LIME [Ribeiro et al., 2016] importance scores as well as a novel, genetically-inspired, black-box attack algorithm, named the `Misinformer` algorithm (see Section 3.2.5). The latter utilises the former in its attack and was shown to be devastating to even the best-performing model trained on the Pheme dataset [Zubiaga et al., 2016]. The `Misinformer` algorithm managed to reduce accuracy to almost that of the most-common-class baseline[1] and the best-performing attack managed to succesfully attack ∼93% of test examples with only a handful of queries to the model. This is evidence that current NLP-based misinformation-detection algorithms can not reliably function outside of the precise datasets they are trained on without work on improving robustness.

The `Misinformer` algorithm was also shown to be resistant to adversarial training, a popular defense technique where example attacks are mixed with the original training set. The models trained with adversarial training retained their accuracy on the unattacked versions of the dataset, but, in attempting to generalise to too wide of a distribution, including both attacks and non-attacks, remained as susceptible to the `Misinformer` but simply required a handful

[1] A baseline that always predicts the most-common class.

more queries of the attacked model per attempt.

The introduction of the `Misinformer` algorithm is an important contribution to the field of misinformation-detection; future work should utilise this algorithm as a benchmark for gauging the impact of releasing a model into the public domain and Authors should aim to contribute defenses that can retain performance even under adversarial scenarios.

5.2 Future work

There are a number of avenues for future work, which, at the time of writing, appear potentially fruitful. It will certainly be challenging to develop robust misinformation-detection models that are resistant to all attacks and, therefore, one possible approach may be to utilise human decision makers in symbiosis with machine learning models, under the pretense that the former is less-susceptible to adversaries. A so-called 'human-in-the-loop' approach, whereby, a model suggests potentially false data-points that are subsequently assessed by a human fact-checker could be a pragmatic solution until fully automated, robust techniques are developed. This may be particularly useful since the model could help to filter out the majority of easy classification decisions and, say, by leveraging uncertainty estimates, only present the human fact-checkers with difficult examples. This could reduce the load on human fact-checkers whilst potentially maintaining a high-level of robustness.

The majority of the susceptibility, at least in the example attacks used, stems from the model's over-reliance on the presence of individual words in assessing veracity, as observed in the potency of the concatenation attacks. In effect, the current formulation is tasking a single model with judging the truth of statements from a wide variety of topics, which, by in large, share no related information. Hence, the root of the issue may be in that a single model cannot effectively generalise to such a wide array of topics without an over-reliance on specific words and thereby techniques to 'split' the model into sub-models per-topic may be effective. An example approach may draw inspiration from meta-learning, for example, the MAML algorithm [Finn et al., 2017], could be used to view misinformation-detection on each separate topic as a distinct learning task. Instead of learning a single, unified model, a shared set-of-weights could encode commonalities between the topics and would be used to fine-tune each model to its given topic. This poses a new issue, which is, the discernment of separate topics. The dataset used in this thesis, Pheme [Zubiaga et al., 2016], provides this information, but this is not necessarily the case, in general, on the internet. Therefore, approaches for topic modelling, such as Latent Dirichlet Allocation [Blei et al., 2003] may be required.

Bibliography

[Alzantot et al., 2019] Alzantot, M., Sharma, Y., Chakraborty, S., Zhang, H., Hsieh, C.-J., and Srivastava, M. (2019). Genattack: Practical black-box attacks with gradient-free optimization.

[Alzantot et al., 2018] Alzantot, M., Sharma, Y., Elgohary, A., Ho, B., Srivastava, M. B., and Chang, K. (2018). Generating natural language adversarial examples. *CoRR*, abs/1804.07998.

[Bannard and Callison-Burch, 2005] Bannard, C. and Callison-Burch, C. (2005). Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, page 597–604, USA. Association for Computational Linguistics.

[Barzilay and Lee, 2003] Barzilay, R. and Lee, L. (2003). Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. *CoRR*, cs.CL/0304006.

[Bellman, 1966] Bellman, R. (1966). Dynamic programming. *Science*, 153(3731):34–37.

[Belohlávek et al., 2018] Belohlávek, P., Plátek, O., Žabokrtský, Z., and Straka, M. (2018). Using adversarial examples in natural language processing. In *LREC*.

[Biggio et al., 2013] Biggio, B., Corona, I., Maiorca, D., Nelson, B., Šrndić, N., Laskov, P., Giacinto, G., and Roli, F. (2013). Evasion attacks against machine learning at test time. *Lecture Notes in Computer Science*, page 387–402.

[Bird et al., 2020] Bird, J. J., Ekárt, A., and Faria, D. R. (2020). Chatbot interaction with artificial intelligence: Human data augmentation with t5 and language transformer ensemble for text classification.

[Blei et al., 2003] Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3(null):993–1022.

[Bozinovski and Fulgosi, 1976] Bozinovski, S. and Fulgosi, A. (1976). The influence of pattern similarity and transfer learning upon training of a base perceptron b2. In *Proceedings of Symposium Informatica*, pages 3–121.

[Breiman, 2001] Breiman, L. (2001). Random forests. *Mach. Learn.*, 45(1):5–32.

[Castillo et al., 2011] Castillo, C., Mendoza, M., and Poblete, B. (2011). Information credibility on twitter. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, page 675–684, New York, NY, USA. Association for Computing Machinery.

[Chen et al., 2017] Chen, Y.-C., Liu, Z.-Y., and Kao, H.-Y. (2017). Ikm at semeval-2017 task 8: Convolutional neural networks for stance detection and rumor verification. In *Proceedings of the 11th international workshop on semantic evaluation (SemEval-2017)*, pages 465–469.

[Cheng et al., 2016] Cheng, J., Dong, L., and Lapata, M. (2016). Long short-term memory-networks for machine reading.

[Cortes and Vapnik, 1995] Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

[Dai and Cai, 2017] Dai, F. Z. and Cai, Z. (2017). Glyph-aware embedding of chinese characters. *CoRR*, abs/1709.00028.

[Derczynski et al., 2017] Derczynski, L., Bontcheva, K., Liakata, M., Procter, R., Hoi, G. W. S., and Zubiaga, A. (2017). Semeval-2017 task 8: Rumoureval: Determining rumour veracity and support for rumours. *arXiv preprint arXiv:1704.05972*.

[Devlin et al., 2019] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding.

[Eger et al., 2019] Eger, S., Şahin, G. G., Rücklé, A., Lee, J.-U., Schulz, C., Mesgar, M., Swarnkar, K., Simpson, E., and Gurevych, I. (2019). Text processing like humans do: Visually attacking and shielding NLP systems. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1634–1647, Minneapolis, Minnesota. Association for Computational Linguistics.

[Enayet and El-Beltagy, 2017] Enayet, O. and El-Beltagy, S. R. (2017). Niletmrg at semeval-2017 task 8: Determining rumour and veracity support for rumours on twitter. In *Proceedings of the 11th international workshop on semantic evaluation (SemEval-2017)*, pages 470–474.

[Finn et al., 2017] Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. *CoRR*, abs/1703.03400.

[Frank and Bouckaert, 2006] Frank, E. and Bouckaert, R. R. (2006). Naive bayes for text classification with unbalanced classes. In *Proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases*, ECMLPKDD'06, page 503–510, Berlin, Heidelberg. Springer-Verlag.

[Freund and Schapire, 1997] Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139.

[Garg and Ramakrishnan, 2020] Garg, S. and Ramakrishnan, G. (2020). BAE: bert-based adversarial examples for text classification. *CoRR*, abs/2004.01970.

[Goodfellow et al., 2015] Goodfellow, I. J., Shlens, J., and Szegedy, C. (2015). Explaining and harnessing adversarial examples.

[Hamid et al., 2020] Hamid, A., Sheikh, N., Said, N., Ahmad, K., Gul, A., Hasan, L., and Al-Fuqaha, A. (2020). Fake news detection in social media using graph neural networks and nlp techniques: A covid-19 use-case.

[Han et al., 2020] Han, Y., Karunasekera, S., and Leckie, C. (2020). Graph neural networks with continual learning for fake news detection from social media.

[Harris, 1954] Harris, Z. S. (1954). Distributional structure. *WORD*, 10(2-3):146–162.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

[Islam et al., 2020] Islam, M. R., Liu, S., Wang, X., and Xu, G. (2020). Deep learning for misinformation detection on online social networks: a survey and new perspectives. *Social Network Analysis and Mining*, 10(1):1–20.

[Iyyer et al., 2018] Iyyer, M., Wieting, J., Gimpel, K., and Zettlemoyer, L. (2018). Adversarial example generation with syntactically controlled paraphrase networks. *CoRR*, abs/1804.06059.

[Jain et al., 2016] Jain, S., Sharma, V., and Kaushal, R. (2016). Towards automated real-time detection of misinformation on twitter. In *2016 International conference on advances in computing, communications and informatics (ICACCI)*, pages 2015–2020. IEEE.

[Jia and Liang, 2017] Jia, R. and Liang, P. (2017). Adversarial examples for evaluating reading comprehension systems. *CoRR*, abs/1707.07328.

[Jin et al., 2019] Jin, D., Jin, Z., Zhou, J. T., and Szolovits, P. (2019). Is BERT really robust? natural language attack on text classification and entailment. *CoRR*, abs/1907.11932.

[Kouzy et al., 2020] Kouzy, R., Abi Jaoude, J., Kraitem, A., El Alam, M. B., Karam, B., Adib, E., Zarka, J., Traboulsi, C., Akl, E. W., and Baddour, K. (2020). Coronavirus goes viral: quantifying the covid-19 misinformation epidemic on twitter. *Cureus*, 12(3).

[Krizhevsky et al., 2014] Krizhevsky, A., Nair, V., and Hinton, G. (2014). The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 55(5).

[Lamb et al., 2021] Lamb, A., Verma, V., Kawaguchi, K., Kannala, J., and Bengio, Y. (2021). Interpolated adversarial training: Achieving robust neural networks without sacrificing too much accuracy.

[LeCun, 1998] LeCun, Y. (1998). The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*.

[LeCun et al., 1989] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551.

[Lee et al., 2021] Lee, N., Li, B. Z., Wang, S., Fung, P., Ma, H., Yih, W.-t., and Khabsa, M. (2021). On unifying misinformation detection. *arXiv preprint arXiv:2104.05243*.

[Li et al., 2020a] Li, D., Zhang, Y., Peng, H., Chen, L., Brockett, C., Sun, M., and Dolan, B. (2020a). Contextualized perturbation for textual adversarial attack. *CoRR*, abs/2009.07502.

[Li et al., 2020b] Li, E., Su, J., Sheng, H., and Wai, L. (2020b). Agent zero: Zero-shot automatic multiple-choice question generation for skill assessments.

[Li et al., 2020c] Li, L., Ma, R., Guo, Q., Xue, X., and Qiu, X. (2020c). BERT-ATTACK: adversarial attack against BERT using BERT. *CoRR*, abs/2004.09984.

[Littlestone, 1987] Littlestone, N. (1987). Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. In *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*, pages 68–77.

[Liu et al., 2017] Liu, F., Lu, H., Lo, C., and Neubig, G. (2017). Learning character-level compositionality with visual features. *CoRR*, abs/1704.04859.

[Liu et al., 2020] Liu, X., Cheng, H., He, P., Chen, W., Wang, Y., Poon, H., and Gao, J. (2020). Adversarial training for large neural language models.

[Ma et al., 2017] Ma, J., Gao, W., and Wong, K.-F. (2017). Detect rumors in microblog posts using propagation structure via kernel learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 708–717, Vancouver, Canada. Association for Computational Linguistics.

[Ma et al., 2018] Ma, J., Gao, W., and Wong, K.-F. (2018). Rumor detection on Twitter with tree-structured recursive neural networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1980–1989, Melbourne, Australia. Association for Computational Linguistics.

[Madry et al., 2019] Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. (2019). Towards deep learning models resistant to adversarial attacks.

[Maini et al., 2020] Maini, P., Wong, E., and Kolter, J. Z. (2020). Adversarial robustness against the union of multiple perturbation models.

[Malouf, 2002] Malouf, R. (2002). A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the 6th Conference on Natural Language Learning - Volume 20*, COLING-02, page 1–7, USA. Association for Computational Linguistics.

[Mikolov et al., 2013] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space.

[Monti et al., 2019] Monti, F., Frasca, F., Eynard, D., Mannion, D., and Bronstein, M. M. (2019). Fake news detection on social media using geometric deep learning.

[Nighojkar and Licato, 2021] Nighojkar, A. and Licato, J. (2021). Improving paraphrase detection with the adversarial paraphrasing task. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7106–7116, Online. Association for Computational Linguistics.

[Niu et al., 2020] Niu, T., Yavuz, S., Zhou, Y., Wang, H., Keskar, N. S., and Xiong, C. (2020). Unsupervised paraphrase generation via dynamic blocking.

[Pang et al., 2021] Pang, T., Yang, X., Dong, Y., Su, H., and Zhu, J. (2021). Bag of tricks for adversarial training.

[Pennington et al., 2014] Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

[Prakash et al., 2016] Prakash, A., Hasan, S. A., Lee, K., Datla, V., Qadir, A., Liu, J., and Farri, O. (2016). Neural paraphrase generation with stacked residual lstm networks.

[Pruthi et al., 2019] Pruthi, D., Dhingra, B., and Lipton, Z. C. (2019). Combating adversarial misspellings with robust word recognition. *CoRR*, abs/1905.11268.

[Raffel et al., 2020] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer.

[Raghunathan et al., 2019] Raghunathan, A., Xie, S. M., Yang, F., Duchi, J. C., and Liang, P. (2019). Adversarial training can hurt generalization.

[Ribeiro et al., 2016] Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "why should i trust you?": Explaining the predictions of any classifier.

[Ribeiro et al., 2018] Ribeiro, M. T., Singh, S., and Guestrin, C. (2018). Semantically equivalent adversarial rules for debugging NLP models. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 856–865, Melbourne, Australia. Association for Computational Linguistics.

[Schott et al., 2018] Schott, L., Rauber, J., Bethge, M., and Brendel, W. (2018). Towards the first adversarially robust neural network model on mnist.

[Shannon, 1948] Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423.

[Shimada et al., 2016] Shimada, D., Kotani, R., and Iyatomi, H. (2016). Document classification through image-based character embedding and wildcard training. In *2016 IEEE International Conference on Big Data (Big Data)*, pages 3922–3927, Los Alamitos, CA, USA. IEEE Computer Society.

[Singh et al., 2017] Singh, V., Narayan, S., Akhtar, M. S., Ekbal, A., and Bhattacharyya, P. (2017). Iitp at semeval-2017 task 8: A supervised approach for rumour evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 497–501.

[Song et al., 2021] Song, C., Shu, K., and Wu, B. (2021). Temporally evolving graph neural network for fake news detection. *Information Processing and Management*, 58(6):102712.

[Srivastava et al., 2017] Srivastava, A., Rehm, G., and Schneider, J. M. (2017). Dfki-dkt at semeval-2017 task 8: Rumour detection and classification using cascading heuristics. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 486–490.

[Su et al., 2020] Su, Q., Wan, M., Liu, X., and Huang, C.-R. (2020). Motivations, methods and metrics of misinformation detection: an nlp perspective. *Natural Language Processing Research*, 1:1–13.

[Szegedy et al., 2014] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2014). Intriguing properties of neural networks.

[Thorne et al., 2018] Thorne, J., Vlachos, A., Christodoulopoulos, C., and Mittal, A. (2018). FEVER: a large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, New Orleans, Louisiana. Association for Computational Linguistics.

[van der Linden et al., 2019] van der Linden, I., Haned, H., and Kanoulas, E. (2019). Global aggregations of local explanations for black box models.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

[Vosoughi et al., 2018] Vosoughi, S., Roy, D., and Aral, S. (2018). The spread of true and false news online. *Science*, 359(6380):1146–1151.

[Wang et al., 2017] Wang, F., Lan, M., and Wu, Y. (2017). Ecnu at semeval-2017 task 8: Rumour evaluation using effective features and supervised ensemble models. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 491–496.

[Wu et al., 2015] Wu, K., Yang, S., and Zhu, K. Q. (2015). False rumors detection on sina weibo by propagation structures. In *2015 IEEE 31st International Conference on Data Engineering*, pages 651–662.

[Zanzotto, 2017] Zanzotto, F. M. (2017). Human-in-the-loop artificial intelligence. *CoRR*, abs/1710.08191.

[Zhang et al., 2019] Zhang, Y., Baldridge, J., and He, L. (2019). Paws: Paraphrase adversaries from word scrambling.

[Zubiaga et al., 2016] Zubiaga, A., Wong Sak Hoi, G., Liakata, M., and Procter, R. (2016). Pheme dataset of rumours and non-rumours.