Adversarial Training Objectives for Generative Attacks on Text Classifiers

by **Thomas Paul Roth**

Thesis submitted in fulfilment of the requirements for the degree of
Doctor of Philosophy
under the supervision of Professor Massimo Piccardi

School of Electrical and Data Engineering
Faculty of Engineering and IT
University of Technology Sydney
November 20, 2024

Certificate of Authorship / Originality

I, Thomas Paul Roth, declare that this thesis is submitted in fulfilment of the requirements for the award of Doctor of Philosophy, in the School of Electrical and Data Engineering at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis. This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program.

Signature: Production Note: Signature removed prior to publication.

Date: November 20, 2024

Abstract

In natural language processing, creating textual adversarial examples is challenging. These examples aim to deceive text classifiers into incorrect predictions while maintaining linguistic similarity to genuine inputs. This complexity arises from the need to preserve the original text's fluency, grammaticality, and semantic coherence, which complicates the generation of effective adversarial samples.

This thesis has focused on refining generative models to produce textual adversarial examples, providing an alternative to the prevalent token-modification attacks that transform document tokens using methods such as word insertion and replacement. Generative approaches, which start with a pre-trained sequence-to-sequence (seq2seq) transformer model and fine-tune it to generate adversarial content, offer benefits like efficiency and parallel processing capabilities. Despite these advantages, generative methods remain under-explored and lack established training methodologies. This research addresses this gap by adapting generative models across various scenarios characterised by unique threat models, assumptions, and objectives, aiming to create examples that mislead classifiers while preserving essential textual properties.

In our first black-box scenario, where only the victim model's predictions are known, our approach uses reinforcement learning with a policy gradient algorithm to fine-tune the seq2seq model, using a reward function that encourages adversarial examples and penalising constraint violations. In the next white-box scenario, the attacker has full knowledge of the victim model's architecture and parameters, accessing loss function gradients. Our method proposes an original differentiable objective function with pre-trained models to improve the textual quality of the adversarial examples, while inducing the victim model into an incorrect classification. Token-mapping matrices connect each model, ensuring end-to-end differentiability and allowing integration across vocabularies, overcoming a key limitation of prior work. The final scenario adapts our previous white-box attack to a multilingual context, requiring the generation of adversarial text in the same language as the original. Our approach, successfully demonstrated across five languages, integrates a language-detection model into the training objective and to our knowledge represents the first generative multilingual attack.

In summary, this thesis has demonstrated the effectiveness of our generative methodologies, highlighting their potential to advance textual adversarial example creation and positioning them as a competitive alternative to established token-modification techniques.

Acknowledgements

I would like to acknowledge and thank my supervisory team for their contributions to this thesis. From UTS, Professor Massimo Piccardi (principal supervisor) and Dr. Iñigo Jauregi Unanue; and from CSIRO's Data61, Alsharif Abauddba and Surya Nepal. Without your patient guidance and assistance this thesis would not be possible. I am very grateful.

I would also like to thank the Australian Government and CSIRO's Data61 for funding the work of this thesis.

This thesis is dedicated to everyone who supported me throughout the journey. To my family, to my friends, to my beautiful wife, and finally, to my dogs, who will be the last to rip it apart.

Thomas Paul Roth
November 20, 2024
Sydney, Australia

Published Content

Tom Roth, Yansong Gao, Sharif Abuadbba, Surya Nepal, and Wei Liu. 2023. Token-Modification Adversarial Attacks for Natural Language Processing: A Survey. To appear in *AI Communications*.

[Under Review] **Tom Roth**, Inigo Jauregi Unanue, Sharif Abuadbba and Massimo Piccardi. 2023. A Constraint-Enforcing Reward for Adversarial Attacks on Text Classifiers. Under second review in the *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*.

Tom Roth, Inigo Jauregi Unanue, Sharif Abuadbba and Massimo Piccardi. 2024. XVD: Cross-Vocabulary Differentiable Training for Generative Adversarial Attacks. Accepted for publication in the proceedings of *The 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*.

Tom Roth, Inigo Jauregi Unanue, Sharif Abuadbba and Massimo Piccardi. 2024. A Generative Adversarial Attack for Multilingual Text Classifiers. In *AAAI-24 Workshop on Artificial Intelligence for Cyber Security (AICS 2024)*. Association for the Advancement of Artificial Intelligence.

Contents

1 Introduction **2**
- 1.1 Motivation . . . 3
- 1.2 Research Questions . . . 5
- 1.3 Research Methods . . . 6
- 1.4 Thesis Organisation . . . 7

2 Literature Review **8**
- 2.1 Token-Modification Adversarial Attacks for Natural Language Processing: A Survey . . . 8
- 2.2 Generative Adversarial Attack Preliminaries . . . 31

3 A Constraint-Enforcing Reward for Adversarial Attacks on Text Classifiers **38**
- 3.1 Introduction . . . 38
- 3.2 Related Work . . . 40
- 3.3 Proposed Approach . . . 41
- 3.4 Experimental Setup . . . 44
- 3.5 Results . . . 46
- 3.6 Human Validation . . . 47
- 3.7 Discussion . . . 48
- 3.8 Conclusion . . . 53
- 3.9 Training Details . . . 54
- 3.10 Compared Attacks . . . 54
- 3.11 Human Validation Methodology . . . 54
- 3.12 Examples . . . 57

4 XVD: Cross-Vocabulary Differentiable Training for Generative Adversarial Attacks **61**
- 4.1 Introduction . . . 61
- 4.2 Related Work . . . 63
- 4.3 Proposed Approach . . . 64
- 4.4 Experimental Setup . . . 68
- 4.5 Results . . . 70
- 4.6 Ablation Analysis . . . 71
- 4.7 Ethical Considerations . . . 72
- 4.8 Conclusion . . . 73

5 A Generative Adversarial Attack for Multilingual Text Classifiers **75**
- 5.1 Introduction . . . 75

5.2 Related Work 77
5.3 Proposed Approach 78
5.4 Experimental Setup 82
5.5 Results 84
5.6 Conclusion 87
5.7 Training Details 87
5.8 Limitations 88

6 Conclusion **90**

List of Figures

2.1 Paper 1: Token-modification attack example 9
2.2 Paper 1: Token-modification attack components 11
2.3 Paper 1: Image vs text perturbations 17
2.4 Paper 1: Saliency maps 21

3.1 Paper 2: Examples of successful attacks 39
3.2 Paper 2: Sample generation 42
3.3 Paper 2: Training approach 42
3.4 Paper 2: Attack success rate and diversity of decoding methods 50
3.5 Paper 2: Fluency scores 52
3.6 Paper 2: Human evaluation instructions 57

4.1 Paper 3: Training approach 65
4.2 Paper 3: Performance by KL divergence and diversity coefficients 72
4.3 Paper 3: Performance by number of generated sequences 73

5.1 Paper 4: Training approach 78
5.2 Paper 4: Results 84

List of Tables

2.1 Paper 1: Token granularities . . . 10
2.2 Paper 1: Transformation examples . . . 13
2.3 Paper 1: Phrase-level perturbations . . . 18
2.4 Paper 1: Constraints . . . 24

3.1 Paper 2: Attack success rate . . . 46
3.2 Paper 2: Baselines . . . 47
3.3 Paper 2: Ablation . . . 49
3.4 Paper 2: Successful adversarial examples on TREC . . . 49
3.5 Paper 2: Training behaviour . . . 53
3.6 Paper 2: Hyperparameters . . . 55
3.7 Paper 2: Models . . . 55
3.8 Paper 2: Baselines and their parameters . . . 56
3.9 Paper 2: Examples of generated candidates for Rotten Tomatoes . . . 59
3.10 Paper 2: Examples of generated candidates for Financial Phrasebank . . . 60

4.1 Paper 3: Dataset statistics . . . 68
4.2 Paper 3: Baselines . . . 69
4.3 Paper 3: Successful adversarial examples . . . 74

5.1 Paper 4: Multilingual successful adversarial examples . . . 85
5.2 Paper 4: Removing the semantic similarity component . . . 86
5.3 Paper 4: Removing the language detection component . . . 86
5.4 Paper 4: TSM splits . . . 87
5.5 Paper 4: Hyperparameters . . . 89

Chapter 1

Introduction

Machine learning models learn patterns from data to make decisions or predictions. They are today used extensively in society in fields ranging from healthcare and finance to entertainment and transportation. Machine learning algorithms can curate content on streaming platforms, assist in medical diagnoses, and even enable self-driving cars. However, these systems are not always robust. In particular, machine learning models are susceptible to malicious inputs that are designed to subvert their intended function, such as deceptive alterations to road signs that mislead autonomous vehicles (Eykholt et al., 2018) or manipulative email content designed to bypass spam filters (Kuchipudi et al., 2020). Given that these models are deployed in many safety-critical and high-stakes domains, this vulnerability is a significant risk, and it is important that we develop more robust and secure machine learning systems.

In the domain of image classification, researchers have found that subtle, imperceptible perturbations can drastically mislead models (Szegedy et al., 2014; Goodfellow et al., 2015). This has been termed an *adversarial attack* and has led to the field of adversarial machine learning, which aims to understand the vulnerabilities of machine learning models (called *victim models*) when exposed to intentionally crafted adversarial examples (Biggio and Roli, 2018). It was quickly evident that models could be easily fooled with simple algorithms, such as the Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2015), the Projected Gradient Descent (PGD) attack (Madry et al., 2018), or the Carlini & Wagner attack (Carlini and Wagner, 2017).

Given these findings it was natural to question the robustness of systems that operate on other data types, one of these being text. Natural Language Processing (NLP) is a subfield of artificial intelligence focused on enabling machines to understand, interpret, and generate human language. Challenges in NLP include language modelling, text classification, machine translation, information extraction, paraphrasing, and many more. Recently, the field has surged in popularity, driven by advances in machine learning, a vast increase in the amount of textual data, and a wide range of possibilities made possible by the remarkable success of large language models.

The aim of both image adversarial attacks and text adversarial attacks is to deceive a machine learning model, but the two differ significantly in their characteristics and the challenges they face. For image adversarial attacks, typically small perturbations are added to the pixel values of an image, usually created using a continuous optimisation

method and bound by a l_p norm (e.g. (Goodfellow et al., 2015)). The perturbations are close-to imperceptible to humans but are sufficient to mislead classifiers. Emphasis is usually put on the perturbed image being imperceptible, or on limiting the magnitude of the change. In contrast, text adversarial attacks might involve changing individual words in the sentence, or paraphrasing or rewriting the sentence. The attack space is discrete, non-differentiable, and imperceptible attacks are very difficult. The textual adversary must also usually make sure their generated text is fluent, grammatically correct, and retains the semantic meaning of the original. For these reasons, adversarial attacks in NLP make up their own subfield and have their own definitions of success, distinct from the image domain.

There are many ways to create text adversarial attacks. Perhaps the most intuitive is to manually change the original text until the classifier is misled, as done in Wallace et al. (2019b). While the text should be then coherent and faithful to the original, manual approaches are labour-intensive and do not scale. Another approach is to use predefined syntactic or lexical transformations to generate adversarial examples, such as a set of templates (Ribeiro et al., 2020) or rules derived from successful adversarial examples (Ribeiro et al., 2018). The most popular approach is to reframe the problem so that finding an adversarial example becomes solving a combinatorial optimisation problem, with the steps being a space of allowed token transformations. These methods are powerful, but are also computationally expensive and limited in their expressiveness. The final broad approach class are the methods that train models to generate adversarial examples. This approach has great potential, but has been not well explored in the literature, and it is this approach that is the main research topic of this thesis.

1.1 Motivation

As mentioned above, combinatorial optimisation attacks, also known as *token-modification* attacks, are the current standard approach for creating adversarial examples. Each attack defines an objective (or 'goal') function, an allowed set of transformations (e.g. word insertion and replacement), a search method (e.g. beam search), and a set of constraints (e.g. must pass a spellchecker). These attacks treat a document as an ordered set of tokens. Individual tokens are transformed until the victim's label changes, with the transformations are applied in an order determined by the search method. The set of constraints specify which transformations are valid or invalid. These attacks are simple to understand, but also powerful, and typically give high attack success rates. They are also easy to implement using open source libraries, such as TextAttack (Morris et al., 2020b).

However, token-modification attacks also have drawbacks. The biggest is their speed. Solving an optimisation problem for each example in a dataset is slow, particularly if the search space of the attack is large. In addition, by treating each example individually, these attacks do not require a training or validation set, but also cannot transfer knowledge across examples. Their expressiveness is also limited by the choice of transformations, restricting the diversity and complexity of their possible adversarial examples. These shortcomings have motivated us to explore alternative approaches to the problem.

We have settled on using generative models, which are becoming increasingly used in solving many machine learning tasks. They are also a viable approach for creating adversarial examples. For this task, the generative model aims to learn a direct mapping from

the original example to the adversarial example, bypassing the need for individual token transformations. There are a wide variety of suitable generative model architectures, such as GANs, autoencoders, or transformers, and the difficulty of training will partially depend on the architecture chosen.

The generative attack has a different paradigm to the token-modification attack, and in turn has different strengths and weaknesses. The generative model is first trained on a separate training set, using a self-supervised training process. Then at inference, it generates a number of candidates for each example on the test set, hoping that one of them is a valid adversarial example. The generative approach hopes that the adversarial patterns it learns from the training set will also work on the test set. As the token-modification attacks do not have a training set, and directly optimise on each example on the test set, the generative approaches will often struggle to achieve the same performance. Generative approaches do have a number of other advantages. Firstly, they are quick at inference, only requiring a single forward pass for a batch of examples. Secondly, they only have to query the victim model once per generated candidate: the token-modification attacks require many queries. Thirdly, they are able to generate in parallel multiple diverse adversarial examples per original, which the token-modification approach cannot. Fourthly, they can produce rich and complex transformations, which depend on how the generative model has been trained. The token-modifications approaches are limited by their search space. Fifthly, they can exploit commonalities across the training distribution to influence perturbations. Sixthly, they can vary the decoding method and its settings to easily change the characteristics of the generated adversarial examples. And finally, the training process is also typically very customisable through a number of hyperparameters. Yet, unfortunately, to date there has been limited work that uses generative models to create text adversarial attacks.

Recently, significant advancements have been made across various NLP tasks (e.g. machine translation, question-answering, summarisation) in part because of the transfer learning paradigm, or more specifically, training using a two-stage pre-training and fine-tuning approach. In this paradigm, models are initially pre-trained on a very large corpus to learn generic language features, such as grammatical structure, the meaning of words and subwords, and so on. They are then fine-tuned on the specific NLP task using a much smaller dataset. Typically, the model is able to leverage much of the knowledge acquired during pre-training, and achieves greater performance than by directly training on the smaller task-specific dataset.

With proven success across NLP tasks, it is highly likely this paradigm would also be effective for the task of generating text adversarial examples. The pre-training stage would allow for the generative model to learn a range of rich transformations, such as paraphrasing, summarisation, or style transfer. The model can then be fine-tuned to attack a specific victim model using a self-supervised adversarial objective function. The goal is for the model to maintain its ability to paraphrase, summarise, and so on, while also being an effective adversarial attacker. This approach, which is the main topic of the thesis, lacks established training methodologies and seems (to us) under-explored.

In this thesis, we have aimed to develop effective methodologies for training generative models to perform textual adversarial attacks. We have tackled three distinct adversarial attack scenarios, each with its own set of threat models, assumptions, and attacker goals, and each requiring an individualised training approach. For each, we have started with a

pre-trained sequence-to-sequence (seq2seq) generative transformer model. We have then fine-tuned this model so that it then generates adversarial examples that flip the label of the original example, while also maintaining textual fluency, similarity, adherence to the ground-truth label, and so on. The overall aim of this work is to show that the generative approach is feasible and a viable alternative to the dominant token-modification techniques of the field.

1.2 Research Questions

We have tackled four research questions in this thesis.

The first research question has addressed the cluttered and fractured nature of the field of textual adversarial attacks that use token modifications. The trend in the field has been to present each attack as a unique contribution, obfuscating that the attack is often just a different combination of a number of components. To rectify this, we have produced a survey that identifies and systematically categorises these core components, which has greatly simplified the field. In turn, this has allowed us to identify the limitations of these attacks, simplify the field for new researchers, and also provides a foundation for the remainder of the thesis.

The following research questions have all investigated generative approaches to creating adversarial attacks. More specifically, each has investigated fine-tuning a pre-trained seq2seq transformer so that it then generates adversarial examples against a text classification model, a foundational and widely applicable NLP task where a model categorises text documents into predefined classes. The research questions have differed in that each has assumed a different scenario for the attacker.

In the second research question we have assumed the *black-box* scenario, where the architecture and internals of the victim model are unknown and the model can only access its predictions. This is a challenging scenario for the attacker as they cannot guide their search with gradient information from the victim model, and have to instead use only the predictions. Our approach has been to turn to reinforcement learning and use a policy gradient algorithm to fine-tune the seq2seq model, using a reward function that rewards adversarial examples and penalises constraint violations.

In the third research question we have instead assumed the *white-box scenario*, where the attacker has full knowledge of the victim model's architecture and parameters and can access the gradients of the loss function. Our proposed approach uses a differentiable objective function comprised of a number of pre-trained models that enforce the textual quality of the adversarial example, while at the same time encouraging the victim model into incorrect classifications. To maintain end-to-end differentiability we have connected each pre-trained model with token-mapping matrices. This feature allows for models with any vocabulary to be integrated, which removes significant restrictions of the prior literature.

Finally, motivated by the growth of multilingualism in the NLP landscape, the fourth research question has explored adapting the previous white-box attack to a multilingual scenario. On top of the usual linguistic challenges, the attacker must additionally generate text in the same language as the original example, which can be any one of a number of different languages. The proposed approach has incorporated an additional language-

detection model into the training objective, to ensure that the output of the attacker is in the expected language. It is to our knowledge the first generative multilingual attack, one of the first multilingual adversarial attacks, and has been able to demonstrate success across five different languages.

To summarise the research questions:

1. Can the common components used by token-modification adversarial attacks be systematically catalogued, for consistent cross-attack comparison and to more clearly identify any limitations?
2. Assuming a black-box scenario, can we fine-tune a pre-trained seq2seq model to generate adversarial examples using a policy-gradient reinforcement learning algorithm?
3. In a white-box scenario, can we fine-tune a pre-trained seq2seq model for generating adversarial examples while maintaining end-to-end differentiability? Can the loss function incorporate multiple pre-trained models to preserve linguistic fluency and grammaticality?
4. Can the aforementioned adversarial attack framework be extended to effectively target multilingual victim models, while preserving the original language and text quality?

The first research question has been covered in Section 2.1 and the next three in Chapters 3 to 5.

1.3 Research Methods

For the survey paper in Section 2.1, we have used the following methodology. We first identified the scope of the survey, and then assembled an initial list of papers on the topic, taken primarily from conference proceedings and published journal papers. We then filtered this list, based on relevance and quality, to assemble a final list of papers. Each selected paper was then read and the component parts identified: namely, the goal function, allowable transformations, search method, and constraints. We then synthesised and categorised the components to form our survey. As the publication processes were long, we had to repeat this process a few times to update the survey with more recent publications.

For the empirical research papers we have used a different methodology. We have started by first identifying a relevant and interesting research question, based on our knowledge of the field. Next, to identify the research gap, we have performed a brief literature review of the chosen topic, a step that has also allowed us to identify what the current state-of-the-art approaches are. Next, we have then designed the approach, and then implemented a prototype to test its feasibility. We have then performed an iterative process to refine the approach until it was clear that the method was sound. Finally, once we have reached an acceptable state, we have performed the experiments and evaluated the results using statistical tests. Finally, we have performed ablation studies, written up the paper, and submitted it for publication.

In terms of datasets, we have opted for publicly available datasets that are commonly used in the NLP literature, also verifying that their terms permit academic use. For our

text classification datasets, we have chosen them to cover multiple classification domains, and languages in the case of Chapter 5.

For our software, we have used Python to implement all approaches, and have used PyTorch and PyTorch Lightning as our base deep learning framework. For NLP libraries we have used those from the Hugging Face ecosystem, such as `transformers`, `datasets` and `tokenizers`. We have also heavily used Weights and Biases for experiment tracking and results visualisation. For hardware, we have conducted all experiments on the UTS iHPC computing cluster.

To ensure that our papers were of high quality, we have used a rigorous writing process, with each draft having been reviewed thoroughly by the supervisory team. Once deemed finished, our papers have been submitted to high-ranked conferences and journals suitable for natural language processing papers.

1.3.1 Ethics

All datasets and models used are publicly available and suitable for academic use. Furthermore, we did not perform surveys on humans or collect any new data, and as such ethics approval was not required for this thesis.

1.4 Thesis Organisation

This thesis is organised as follows. The literature review is presented in Chapter 2, and is split into two parts. The first, Section 2.1 addresses the first research question by presenting a detailed survey of adversarial token-modification attacks. The second, Section 2.2, covers additional content useful to understand the remaining research chapters, including transformers, generative adversarial attacks, policy gradients, and multilingual models. The next three chapters, Chapters 3 to 5, contain the completed units of work that address each research question. Finally, Chapter 6 concludes the thesis.

Chapter 2

Literature Review

In this chapter we present the background required to understand the remainder of the thesis. We have split this chapter into two sections. The first section is made up of our published survey paper. This section serves as an broad introduction to text adversarial attacks and answers the first research question of the thesis, described in Section 1.2. The second section presents the additional preliminaries used in generative adversarial attacks, and presents material to help in understanding Chapters 3 to 5.

2.1 Token-Modification Adversarial Attacks for Natural Language Processing: A Survey

Abstract. Many adversarial attacks target natural language processing systems, most of which succeed through modifying the individual tokens of a document. Despite the apparent uniqueness of each of these attacks, fundamentally they are simply a distinct configuration of four components: a goal function, allowable transformations, a search method, and constraints. In this survey, we systematically present the different components used throughout the literature, using an attack-independent framework which allows for easy comparison and categorisation of components. Our work aims to serve as a comprehensive guide for newcomers to the field and to spark targeted research into refining the individual attack components.

2.1.1 Introduction

Adversarial machine learning investigates the susceptibility of machine learning models to manipulations by an adversary through an *adversarial attack* (Kurakin et al., 2017). While the majority of adversarial machine learning research has focused on image processing, there has been rapidly increasing interest in the NLP domain, motivated by the unique challenges and opportunities presented by textual data. In this work, we survey the recent advancements in adversarial attacks within the NLP domain.

An *adversarial example* is a carefully crafted instance created by introducing a perturbation to an *original example*. This perturbation is designed in such a way that the attacked *victim model* processes the original example as expected but fails on the adversarial example in a manner predetermined by the attacker. To illustrate this concept, Wallace et al.

(2020a) have aimed to perturb a document reading *"Save me it's over 100F"* to introduce a specific token *"22C"* into the output of an English-to-German translation model (see Figure 2.1). Their attack algorithm achieved this by replacing *"100F"* with *"102F"*. While the model translates the original example correctly as *"Rette mich, es ist über 100F"*, it now translates the adversarial example as *"Rette mich, es ist über 22C"*, thereby introducing the specific token, and in turn, effectively demonstrating the potential potency of adversarial attacks.

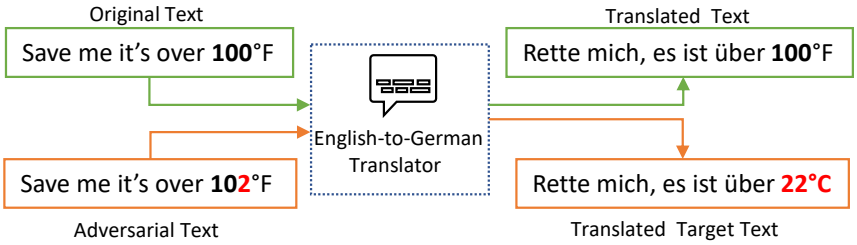


Figure 2.1: Example of a token-modification adversarial attack on a machine translation model.

Usually text adversarial examples must also meet a set of constraints. For example, one constraint might specify that the semantic meaning of the original example is maintained. In this case, *"Save me it's above 100F"* would be valid, but not the proposed *"Save me it's over 102F"*. A second constraint might require that the adversarial example appear non-suspicious to a human. This rules out *"Save MySeLFFFFF it's xxoverxx 100F"*, but our example *"Save me it's over 102F"* would pass.

Many methods have been developed to create text adversarial examples, with the literature predominantly focusing on what we term **token-modification** attacks. These attacks strategically modify individual document tokens at various granularities (e.g., words, characters) to achieve the desired adversarial effect. (Our above example illustrates a token-modification attack, and Table 2.1 gives more examples.) In these attacks, the adversary maintains the syntactic structure of the document, unlike, say, a paraphrase or style-transfer model. Token-modification attacks treat the document as an ordered set of tokens and systematically apply *transformations* to individual tokens. These transformations can include operations such as replacing a word with its synonym, deleting a character, or inserting a phrase into the document. The attack process continues iteratively until the adversary achieves its predefined goal or until a specific stopping criterion is met. This results in a controlled and targeted manipulation of the text that can "fool" the victim model.

Previous surveys, such as those conducted by Zhang et al. (2020) and Alshemali and Kalita (2020), categorise text attacks based on various criteria: token granularity, adversary goal, model access, targeted NLP task, and so forth. While this categorisation scheme provides a structured overview, it tends to over-emphasise the differences between attacks. In reality, many attack algorithms may only differ in their use of a single component, leading to a situation where algorithms can shift between categories with only trivial modifications. Recognising this limitation, in this survey, we adopt a novel perspective by categorising based on the individual attack components, rather than the attacks themselves. This approach offers a more nuanced understanding of the underlying mechanisms and commonalities across different attacks. To the best of our knowledge, we are the first survey to adopt this component-centric approach.

Granularity	Example
Original	*The movie was absolutely fantastic and I loved it!*
Character-level	*The movie was absolutely f4ntastic and I loved it!*
Subword-level	*The movie was absolutely fantastical and I loved it!*
Word-level	*The film was absolutely fantastic and I loved it!*
Phrase-level	*The cinematic experience was absolutely fantastic and I loved it!*

Table 2.1: Another example of a token-modification adversarial attack on a sentiment classifier. The original example is categorised as positive, and the adversarial attack attempts to change the predicted label to negative through changing its tokens, which can be at a range of granularities.

2.1.2 Attack framework

Recently, Morris et al. (2020b) introduced a comprehensive framework that defines four essential components of a textual adversarial attack: (a) a goal function, which determines the success criteria of the attack; (b) a set of transformations, outlining the specific actions that the adversary can undertake; (c) a search method, dictating the sequence in which transformations are applied; and (d) a set of constraints, establishing the boundaries for a valid adversarial example. This formulation casts each text attack as a constrained search problem, defined by its unique combination of these components.

Inspired by the work of Morris et al. (2020b), who also used their framework to develop a software library for rapidly implementing text adversarial attacks, we extend this framework and conduct a systematic survey of the various attack components. Our focus is specifically on token-modification attacks, as they represent the predominant attack type in the current landscape. While the framework could theoretically be extended to other attack classes, its application is not straightforward for methods such as those that train a model to generate adversarial examples. Therefore, we restrict our survey to this prevalent and well-defined category of attacks. Our categorisation, along with an exploration of the range and nuances of attack components, is summarised in Figure 2.2.

Victim model architectures. The model targeted by an adversarial attack is called the victim model, which in this context are models that perform various NLP tasks. The majority of modern NLP models follow a neural network-based architecture, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), long short-term memory networks (LSTMs), and transformers. This class of models currently achieve the state-of-the-art performance across many NLP tasks and are hence widely used.

While many adversarial attacks are agnostic to the model's architecture, some techniques assume the victim possesses specific architectural features, such as embedding layers, attention modules, or gradient information. Simpler models, such as a bag-of-words classifier, might not possess these features and the attack will not work. However, it's arguable that these simpler models also inherently possess a multitude of 'adversarial' examples, due to their basic nature and lower performance on legitimate inputs. As such, the assumption is usually made that the victim models targeted are neural network-based.

Notation. Let Ex be the original example with label y_{true}. All examples are documents, and each document is made up of a set of tokens $\{t_1, \cdots, t_n\}$, for which a token t_i has the numeric representation $\boldsymbol{e}_{t_i}$ (e.g. word embeddings). The document Ex is transformed to a numeric representation $\boldsymbol{x}$ and then acted on by a model f. Likewise, the adversarial

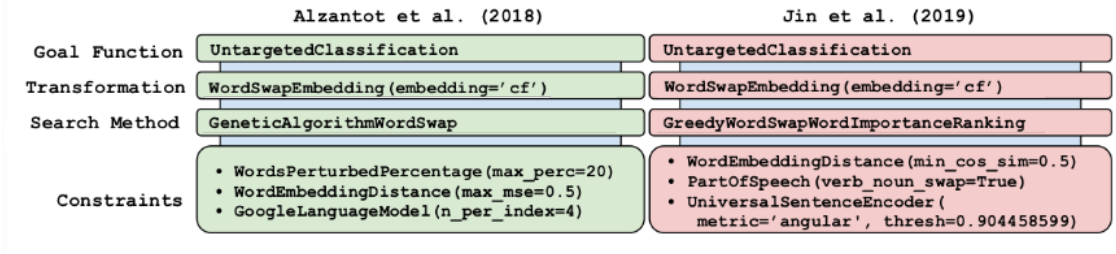


Figure 2.2: This example, from the TextAttack (Morris et al., 2020b) documentation, neatly demonstrates how token-modification attacks (Alzantot et al. (2020) and (Jin et al., 2020) here) are made up of a combination of four components. For an easy comparison, the TextAttack paper (Morris et al., 2020b) also provides a table that summarises the combinations of a number of popular attacks. We cover the individual components in detail in Sections 2.1.3, 2.1.4, 2.1.5, and 2.1.6.

example is Ex' (also with label y_{true}), its numeric representation is $\boldsymbol{x}'$, and we denote its output $f(\boldsymbol{x}')$ by y'. We use l to represent a generic loss function that acts on a single example.

2.1.3 Adversary goals

The adversary goal defines what the adversary aims to achieve. Many tasks in NLP, ranging from sentiment analysis and spam detection to more complex tasks like question answering and machine translation, can be broadly classified as having either classification goals or seq2seq goals, depending on the task the victim model addresses.

2.1.3.1 Classification

Attacks on text classifiers, which include tasks such as document categorisation and sentiment analysis, can be broadly considered as either **targeted** or **untargeted** attacks. Targeted attacks attempt to induce the model to classify Ex' as a specific class y_{target} different from y_{true} by minimising $l(f(\boldsymbol{x}'), y_{target})$. Untargeted attacks only require that Ex is misclassified, and their algorithms typically aim to maximise $l(f(\boldsymbol{x}'), y_{true})$. The two are equivalent for binary classification, and for multiclass classification, converting between the two is usually straightforward.[1]

There also exist alternate goal functions. For example, Boucher et al. (2022) propose an *availability attack* goal function that aims to maximise the model processing time.

2.1.3.2 Sequence-to-sequence

The seq2seq tasks in NLP cover a wide spectrum, including machine translation, text summarisation, paraphrase generation, speech recognition, speech synthesis, and image generation from captions. These tasks often involve complex transformations and mappings between input and output sequences. Unlike classification tasks, where the adversary goals are more standardised, seq2seq tasks do not have a uniform set of adversary goals. Consequently, we adopt a broad classification for seq2seq attacks, categorising them as

[1] For example, instead of perturbing $\boldsymbol{x}$ towards a *specific* decision boundary, an untargeted attack perturbs $\boldsymbol{x}$ towards the *nearest* decision boundary.

either untargeted or targeted, to capture the range of possible adversarial objectives within this domain.

Untargeted attacks: Untargeted attacks in the seq2seq context aim to degrade the quality of the output sequence as much as possible, as measured by a specific metric. The objective is to disrupt the model's performance without necessarily guiding the output towards a particular target. This form of attack is prevalent in seq2seq tasks and has been employed in various contexts (e.g. Hsieh et al., 2019; Michel et al., 2019; Camburu et al., 2020; Jia and Liang, 2017; Ebrahimi et al., 2018a; Wan et al., 2022).

Targeted attacks: Targeted attacks, on the other hand, seek to induce specific alterations in the output sequence. The goals can be diverse, ranging from changing a particular output token to any other token (Papernot et al., 2016; Ebrahimi et al., 2018a; Zhao et al., 2018), altering every output token from its original value (Cheng et al., 2020), modifying a set of specific output tokens to another set of specific tokens (Wallace et al., 2020a; Ebrahimi et al., 2018a), to introducing a set of specific output tokens anywhere in the output sequence (Hsieh et al., 2019; Cheng et al., 2020; Zhao et al., 2018), or even finding an incorrect input sequence that generates a specific output sequence (Wallace et al., 2020a; Zhang et al., 2021). These attacks necessitate the careful formulation of an attack-dependent loss function, which is then optimised to either maximise or minimise, depending on the specific adversarial goal.

2.1.4 Transformations

A transformation is a function that alters one ordered set of tokens into another. These transformations can be grouped into several broad categories, referred to here as *transformation types*. Common transformation types include replacing one token with another, inserting a token, deleting a token, changing token ordering, and merging two tokens (e.g. Li et al., 2021). Among these, replacing a token is the most prevalent transformation type, and thus, we give it special emphasis in the following discussion.

When executing a token replacement, where a token t is substituted with a replacement t_r, the attack algorithm must select t_r from a *set of replacements*, denoted by $\mathbb{S}(t)$, with the parameterisation indicating its dependence on t. The method for constructing $\mathbb{S}(t)$ is a core component of many attack algorithms. Typically, the process involves generating an initial set of candidates and then applying constraints to filter the set. For instance, an algorithm might use a dictionary to generate a set of synonyms for t and then filter out candidates that do not share the same part-of-speech (POS) tag with t.

Below, we categorise methods that have been employed to generate the initial set of replacement candidates, while deferring the discussion of filtering constraints to Section 2.1.6. Attacks can, and do, employ a combination of these methods, and have devised a wide range of nuanced strategies to effectively deceive the victim model.

2.1.4.1 Any replacement

The adversary may replace token t with any token of the same granularity. While this approach is straightforward and can be highly effective in degrading the victim model's performance, it often results in ungrammatical constructions and loss of semantic meaning. Effectively, these attacks are prioritising minimising the perturbation size without

Transform type	Example	Sample references
Any token replacement	*dog → !og*	Ebrahimi et al. (2018b,a); Gao et al. (2018); Formento et al. (2023)
Adjacent keyboard typos	*achieve → axhieve*	Belinkov and Bisk (2018); Li et al. (2019)
Common misspellings	*achieve → acheive*	Belinkov and Bisk (2018)
Number to word form	*12 → twelve*	Mondal (2021)
Leetspeak conversions	*leet → l33t*	Emmery et al. (2021)
Vowel removal, letter shuffling, etc	*letters → ltetrs*	Eger and Benz (2020)
Phonetic replacements	*fair → fare*	Le et al. (2022); Liu et al. (2023)
Homograph adjective manipulation	*short lead → hot lead*	Emelin et al. (2020)
Unicode-based replacements	*a → à*	Eger et al. (2019)
Visual character mapping	*cl → d*	Li et al. (2020a); Dionysiou and Athanasopoulos (2021)
Invisible characters	*abc → abc¡U+200B¿*	Boucher et al. (2022)
Inflectional perturbations	*run → running*	Tan et al. (2020)
Thesaurus-based synonyms	*happy → joyful*	Ren et al. (2019); Fellbaum (1998); Zang et al. (2020)
Word to emoji	*love → ♡*	Chang et al. (2023)
Word embedding-based	*fast → swift*	Pennington et al. (2014); Li et al. (2019); Xu and Du (2020)
Counter-fitted synonyms	*happy → content*	Mrkšić et al. (2016); Alzantot et al. (2020)
Context-aware embeddings	*bank → credit union*	Li et al. (2020b)
Masked language model infill	*go [MASK](here) fast → go very fast*	Alzantot et al. (2020); Zhang et al. (2019); Li et al. (2020b); Garg and Ramakrishnan (2020); Li et al. (2021); Lei et al. (2022)
Phrase replacements	*quickly run → swiftly sprint*	Lei et al. (2022); Shao et al. (2022); Chang et al. (2023)
Predefined parse template filling	*[Noun] runs [Adverb] → The dog runs quickly*	Shao et al. (2022)
Derived rule-based	*What NOUN → Which NOUN*	Yuan et al. (2021a)

Table 2.2: Examples of different transformation types, and some examples of methods where they were used.

attempting to maintain semantics, and consequently, this method is rarely employed, except in character-level attacks that permit any alphanumeric character (upper or lower case) or punctuation as a valid replacement (Ebrahimi et al., 2018b,a; Gao et al., 2018; Formento et al., 2023).

2.1.4.2 Human errors

The adversary may replace token t with a token that simulates an innocent human-made mistake, such as a typographical error. These transformations leverage the natural imperfections in human writing to craft adversarial examples that blend seamlessly into genuine text, making detection more challenging.

Several transformation types fall under this category:

- *Typos:* Mimicking keyboard typos by replacing letters with adjacent characters on a QWERTY keyboard (Belinkov and Bisk, 2018; Li et al., 2019).
- *Common misspellings:* Substituting a word with a commonly-misspelled variant, using resources like Wikipedia or corpora tracking document edit history (Belinkov and Bisk, 2018).
- *Number expansion:* Expanding numbers into their word form (Mondal, 2021).
- *Leetspeak:* Converting characters to their "leetspeak" equivalents (Emmery et al., 2021).
- *Other simple transformations:* Examples include removing vowels, shuffling letters, inserting symbols, joining or truncating words, and using phonetic or visually-similar words (Eger and Benz, 2020).
- *Phonetic similarity:* Choosing replacements that are phonetically similar. This can be done with algorithms like the SOUNDEX algorithm (Stephenson, 1980) or using the Pinyin equivalents for Chinese characters (Le et al., 2022; Liu et al., 2023).
- *Homograph adjectives:* Homographs are words with the same spelling as another but with a different meaning. Emelin et al. (2020) constructed an attack where they inserted or replaced the adjectives in front of homographs, with the intent to mimic word sense ambiguity errors.

Table 2.2 provides examples and references for each of these transformation types.

2.1.4.3 Visually similar

The adversary may replace token t with a visually similar token, intending that a human reader overlooks the discrepancy. These transformations can be applied at the character level, including many-to-one or one-to-many character replacements (e.g., replacing *cl* with *d*). The effectiveness of these attacks may vary depending on the font used and the specific replacements chosen.

Several methods have been proposed to achieve visually similar transformations:

- *Manual mapping:* Creating a mapping between each token in the vocabulary and a manually chosen replacement (Eger et al., 2019; Li et al., 2019). For instance, characters may be replaced with versions that include diacritics (e.g., *c* to *č*) (Eger et al., 2019). Unicode descriptions of characters can also be used to find replacements of the same character and case. In languages like Chinese, characters may be replaced with those sharing partial pictographs (Liu et al., 2023).
- *Visual embeddings:* Representing each character as a grid of pixels and numerically encoding them, either with raw pixel vectors (Eger et al., 2019) or dense representations from the intermediate layers of a CNN (e.g. Li et al., 2020a; Dionysiou and Athanasopoulos, 2021). The set of replacements $\mathbb{S}(t)$ can be constructed by selecting the closest points to the character in the latent space.
- *Imperceptible characters:* Designing attacks that are truly imperceptible to the human eye by using invisible characters, homoglyphs, reordering characters, or deletion characters (Boucher et al., 2022).

These visually similar transformations present a unique challenge in adversarial attacks, as they exploit the human tendency to overlook subtle visual discrepancies. The diversity of methods employed reflects the complexity of crafting attacks that are both effective against machine learning models and inconspicuous to human readers.

2.1.4.4 Synonyms

Adversarial attacks may replace a token t with a synonym that maintains the same meaning. This can be achieved through various methods, including using predefined synonym lists or pre-trained token embedding spaces. Both approaches aim to maintain the semantic content of the original text, while keeping perturbations small and local.

Predefined list The adversary replaces t with a replacement t_r taken from a predefined list of tokens that share the same meaning. Various lists can be used:

- Thesaurus, lexical database, or knowledge database, such as WordNet (Fellbaum, 1998; Ren et al., 2019), HowNet (Dong et al., 2010; Zang et al., 2020), or DrugBank (Wishart et al., 2018) for biomedical data (Mondal, 2021).
- Inflectional perturbations of words with the same base meaning (Tan et al., 2020). Inflectional perturbations modify a word's form while retaining its core meaning, such as changing *run* to *running*.
- Emoji equivalents for words (Chang et al., 2023).

This method often struggles to distinguish the sense in which a word is used, leading to replacements that may read strangely or not fit well in context. Some attacks add constraints to mitigate this issue, but the problem often persists (e.g. Zang et al., 2020).

Embedding space The adversary replaces token t with a token t_r corresponding to a nearby point in a token embedding space. This approach assumes that tokens with similar meanings cluster together in embedding space, forming a natural set of synonyms. Different embedding spaces can be used:

- Using word embeddings like GloVe (Pennington et al., 2014; Li et al., 2019; Xu and Du, 2020), though this may include related words like antonyms or words with similar concepts but different meanings. Example of antonyms (e.g. *east* and *west*, or *fast* and *slow*) or words with similar concepts but different meanings (e.g. *English*, *American*, and *Australian*).
- Using *counter-fitted* embeddings (Mrkšić et al., 2016) as a post-processing step for word embeddings. This technique pushes together similar words while also pushing apart related words, resulting in an embedding space where proximity indicates synonymy. It enhances the precision of synonym selection in adversarial attacks (Alzantot et al., 2020; Jin et al., 2020; Wang et al., 2021a; Li et al., 2020b).
- Considering contextual embeddings such as BERT (Devlin et al., 2019) to provide context-aware token representations. Traditional token embeddings represent a token with a single point, leading to replacements that may not match the token's sense in a specific context. Contextual embeddings like BERT mitigate this issue by considering the surrounding text, enabling more accurate and contextually relevant replacements (Hsieh et al., 2019).

Different methods have been used to define "closeness" in embedding space, to identify the set of replacements. Techniques have included:

- *K-nearest neighbours (KNN):* This approach identifies the k closest tokens to the original in the embedding space, based on distance metrics such as Euclidean distance or cosine similarity (e.g. Li et al., 2019; Xu and Du, 2020).
- *Distance thresholding:* Tokens within a specific distance δ to the original token are considered. The distance can be measured using various metrics, providing flexibility in defining semantic closeness (e.g. Zhang et al., 2019).
- *Hybrid approach:* Methods can combine the KNN and distance thresholding methods, attempting to ensure that the chosen substitutes are both close and relevant (e.g. Alzantot et al., 2020; Jin et al., 2020; Hsieh et al., 2019).
- *Synonym clustering:* Creating clusters of synonyms in the embedding space allows for the replacement of a "central" word with any word in the cluster (Wang et al., 2021a).
- *Continuous algorithms:* In many image-based adversarial attacks, the algorithm iteratively perturbs the input image in a continuous space to cross the decision boundary of the classifier, effectively fooling the model. In the text domain, this concept can be adapted to select the synonyms that move the adversarial example closer to the victim model's decision boundary in the embedding space. For example, Meng and Wattenhofer (2020) adapted the DeepFool algorithm (Moosavi-Dezfooli et al., 2016) for this purpose. Generally, without modification these methods struggle to maintain semantic coherence Liang et al. (2018), as Figure 2.3 illustrates. Wang et al. (2020a) attempt to circumvent this by mapping the text to continuous space and back with tree-based autoencoders and decoders.

2.1.4.5 Language model predictions

Language models (LMs) can be used by adversaries to find replacements for tokens, often through *masked language modeling*. The typical process involves replacing the target token with a mask, obtaining predictions from the LM, and using the top predictions as the set of replacements. Variations on this approach are common (Alzantot et al., 2020; Zhang et al., 2019; Li et al., 2020b; Garg and Ramakrishnan, 2020; Li et al., 2021; Lei et al., 2022). For example, consider the sentence "*The technology accelerates rapidly*". The token *accelerates* is masked and then replaced with a top LM prediction, such as *evolves*, resulting in "*The technology evolves rapidly*". The advantages of this approach class include higher-quality replacements, as context is considered, and a tendency to avoid rare tokens, which appear suspicious (Ippolito et al., 2020). Drawbacks include the need for complex and memory-intensive LMs, slower replacement generation, and the inability to cache synonyms between documents (as they depend on document context). Some approaches mitigate these by combining both LMs and word embeddings (Alzantot et al., 2020; Zhang et al., 2019).

Related techniques have included occlusion and language models (OLM) (Harbecke and Alt, 2020); a method from the interpretability literature that identifies relevant words to replace (Malik et al., 2021), or Dropout Substitution (Zhou et al., 2019); a technique that applies dropout to internal embeddings of target words to enhance candidate quality

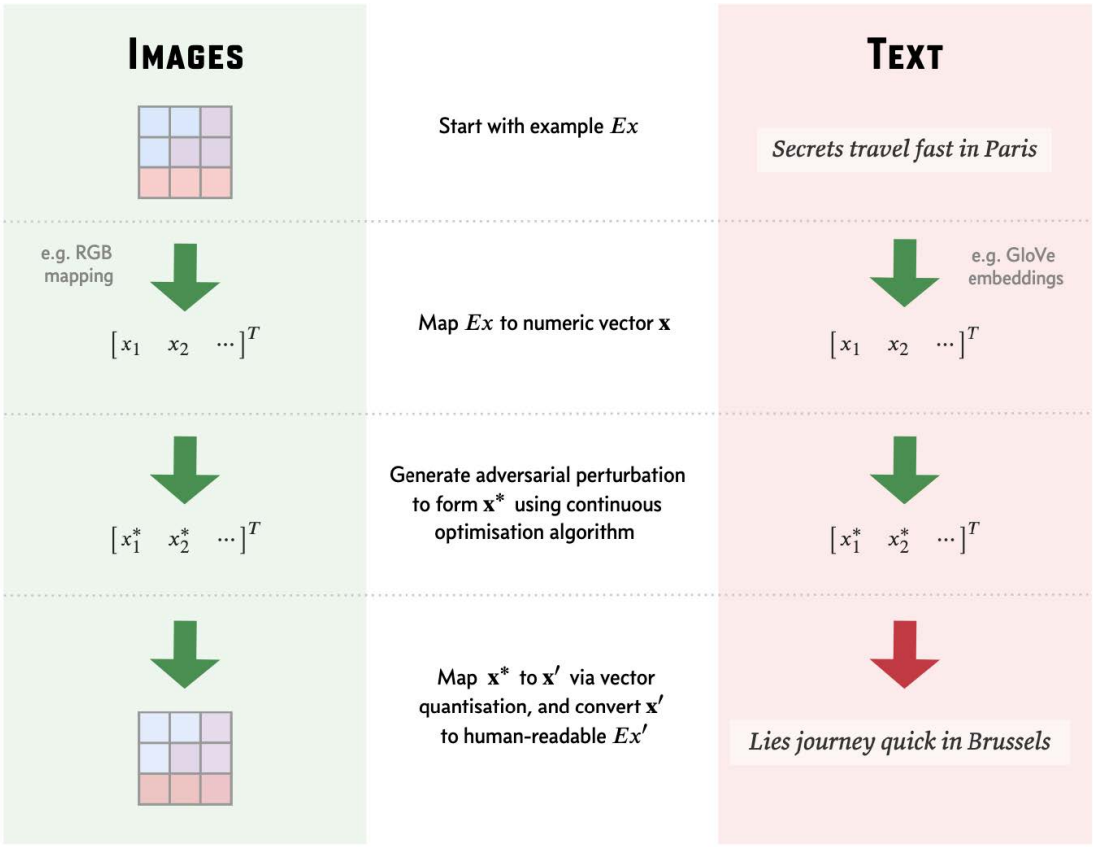


Figure 2.3: Continuous perturbations in a numerical space work for image adversarial attacks, where the human eye cannot pick up subtle colour changes, but in the text domain, will be very conspicuous. In other words: while $\boldsymbol{x}^*$ fools the model, its text form Ex' may not be semantically valid. The red arrow shows the problematic step.

(Emmery et al., 2021).

2.1.4.6 Phrase manipulations

Attacks have been developed that identify and manipulate whole phrases, rather than tokens. The motivation is to create more linguistically diverse and rich adversarial examples than is possible through a series of individual token-level transforms. We show some examples in Table 2.3.

To add a phrase into the sentence, a number of approaches can be taken. One option is to make use of a parse template, which in this context can be thought of as a predefined format or structure for a sentence. For example, say the adversary wishes to add an adverb phrase to a sentence. The adversary would first identify the examples that it would make sense to add an adverb phrase too, perhaps by comparing the example to the template. Then the adversary would identify an adverb phrase that makes sense in the sentence context, perhaps by using a language model. This is the method used by (Shao et al., 2022).

For a controllable phrase replacement process, Lei et al. (2022) propose using two stages to the attack process: identifying phrases in the original sentence suitable to attack, and proposing adversarial replacements that still keep the textual quality.

To identify the phrase chunks from the original sentence, one approach is to use a syntactic parser, such as the Stanford Parser (Manning et al., 2014). Parsers like this one identify phrases by first tokenising the text and then tagging each token with its part-of-speech (e.g. noun, verb), dependent on its role in the sentence. It then applies a parsing algorithm to construct a hierarchical syntactic tree that represents the grammatical structure of the sentence. From this tree, phrases can be identified. To identify which phrase to attack, any of the methods in Section 2.1.5 can be used.

To find a suitable replacement phrase, a number of methods can be used. A simple approach is to replace individual words with their dictionary definitions, maintaining the same part-of-speech (Chang et al., 2023). While this does add a phrase in, it is easy to introduce a phrase that does not make sense in the overall sentence context. A second approach is to use the mask-infill models described in Section 2.1.4.5, where the phrase is masked and then infilled by a language model. The language model chooses the replacement to best suit the context that it is used in the original sentence (Lei et al., 2022).

Manipulation and Method	**Example**
Add a phrase using parse template and language model	Original: "*Birds fly.*" Template: "*Birds fly [ADVP].*" Adversarial: "*Birds fly quickly and quietly.*"
Replace a phrase with its dictionary definition	Original: "*White beaches surrounded by azure seas.*" Definition: "azure: bright blue in color like a cloudless sky" Adversarial: "*White beaches surrounded by bright blue in color like a cloudless sky seas*"
Replace a phrase using a mask-infill model	Original: "*The cat sat on the mat.*" Masked infill: "*The cat [MASK] the mat.*" Adversarial: "*The cat lay down on the mat.*"

Table 2.3: Examples of some phrase-level manipulation techniques.

2.1.4.7 Set of rules

One broad approach is to search for and use a set of simple find-and-replace rules that, when applied, successfully create adversarial examples for many examples in a dataset. These global rules, derived from patterns found among successful adversarial examples, are simple and effective in attack strength, but will struggle to keep textual fluency and readability in many cases. Ribeiro et al. (2018) is one work that explores this approach. They select rules that maximise semantic similarity and have broad applicability. Examples of their rules include replacing *?* with *??*, or changing the words around a tagged part-of-speech chunk, such as *What [NOUN]* to *Which [NOUN]*. Extending this idea, Yuan et al. (2021a) crafted word replacement rules by identifying adversarial example patterns effective against many different types of models. These replacements, though simple, were able to fool 63 victim models at once. Examples of these word replacement rules include *keep* to *preserve*, *President* to *Chairman*, *executives* to *administrators*, and so on.

2.1.4.8 Homographs

Emelin et al. (2020) have devised an attack that makes use of homographs in the document. Homographs are words that are spelled the same, but have different meanings in different contexts, and also might be pronounced differently. This particular attack targets the words in front of the homograph to subtly change the context of the sentence. For example, the attack might change "*They were discussing a strike in the meeting.*" to "*They were discussing a lightning strike in the meeting.*" Originally, the sentence might suggest a discussion about a labour protest, but the addition of *lightning* changes the context to a weather-related incident. As not every sentence contains targetable homographs in this way, the attack will carefully select examples they can attack, and will leave the rest.

2.1.5 Search methods

The search method, also referred to as the optimisation method, aims to identify an effective sequence of transformations that satisfies the given constraints of the adversarial problem. The search process terminates either upon achieving the adversarial goal, after a specified number of queries, or when further exploration is restricted by the constraints.

Search methods typically rank each transformation with a **scoring function**, also known as an objective or fitness function, that evaluates the quality of the solution in the search space. In the context of adversarial attacks, the scoring function quantifies or estimates the effect of the transformation on the adversarial loss function.

2.1.5.1 Preliminaries

Gradient approximations The simplest scoring function calculates the increase in loss directly due to the transformation. While exact, this method is computationally expensive because it requires a forward and backward pass through the model for each candidate transformation, which becomes prohibitive for large models.

A widely-used alternative is to use a *gradient-based* scoring function (e.g. Yoo and Qi, 2021; Berger et al., 2021), which employs the gradient of the loss with respect to the input, denoted as $\partial l / \partial \boldsymbol{x}$. This gradient serves as a first-order approximation of how the loss l changes when elements of $\boldsymbol{x}$ are altered.[2] The transformations are then ranked based on their dot product with the gradient, $\partial l / \partial \boldsymbol{x} \cdot (g(\boldsymbol{x}) - \boldsymbol{x})$, where $g(\boldsymbol{x}) - \boldsymbol{x}$ represents the relative change in the input due to the transformation g.

This gradient-based approach is faster, requiring only a single backward pass, but it assumes the adversary has access to the model parameters (*white-box* attack). It also rests on two assumptions: that the loss function l is nearly linear around $\boldsymbol{x}$, and that other features remain constant when evaluating a single feature's importance (Singla et al., 2019). While these assumptions may not always hold, the method is generally effective in finding successful adversarial examples.

As a compromise, Wallace et al. (2020a) use a hybrid approach which has advantages of both approaches: the speed of the gradient approximation, and the accuracy of direct-checking. They first use the gradient approximation to create a shortlist of the top k

[2] Some attacks (e.g. Papernot et al., 2016) instead use the Jacobian matrix $\partial f(x) / \partial \boldsymbol{x}$ and directly change model predictions.

transformations across all tokens in Ex. They then find the best transformation by directly checking the loss change of each.

Proxy models In black-box adversarial attacks, where direct access to the target model's architecture or parameters is unavailable, attacks may use proxy models, also known as substitute or replacement models, to aid their attack. The proxy model is intended to mimic the characteristics of the target model, including its gradients, predictions, and confidence values. The attacker usually trains a proxy model on the same dataset as the victim model, or at least, a dataset from a similar domain. The attack can then use the proxy model's gradients or outputs to identify promising replacements for a token, or select where to attack next. This approach has been used in various works, such as (Papernot et al., 2017; Zhu et al., 2022; Yuan et al., 2023).

2.1.5.2 Algorithms

A variety of optimisation algorithms have been proposed in the literature, which we categorise and present below.

Simple heuristics The most straightforward is to apply transforms according to a simple heuristic. This is simple to understand but may not find as good a solution as a more advanced algorithm. Some heuristics include

Batch application: This method applies a transformation to all eligible tokens in the example Ex at once. Employed by Tan et al. (2020) and Ebrahimi et al. (2018a), it replaces each applicable word $w \in Ex$. Variants include multiple runs with each selecting a different transformation (Belinkov and Bisk, 2018) or replacing each token with a probability p (Eger et al., 2019).

Sequential application: Tokens are transformed in the order they appear in the document. The algorithm iterates through document tokens from start to finish and applying the best transformation for each token until the attack succeeds, cycling if necessary. Zhang et al. (2019) use this method but select transformations based on predefined probabilities, rather than selecting always the best one.

Random application: Tokens are randomly selected and transformed. Used as a baseline by Yoo et al. (2020) and Hsieh et al. (2019), this method replaces tokens either with the best or a random replacement.

Importance-based methods This approach ranks tokens based on their influence on the loss function, transforming the most impactful tokens first to minimise the perturbation size. Yoo et al. (2020) term this an *importance ranking*. Saliency maps can be an effective way to visualise these importance rankings, as shown in Figure 2.4.

There are various ways to construct an importance ranking.

- *Gradient-based ranking:* The gradient magnitude $\|\partial l / \partial \boldsymbol{e}_{t_i}\|_2$ serves as a heuristic for token importance. While this method doesn't capture the direction of the gradient, it remains a widely-used heuristic (Li et al., 2019; Xu and Du, 2020; Samanta and Mehta, 2017).

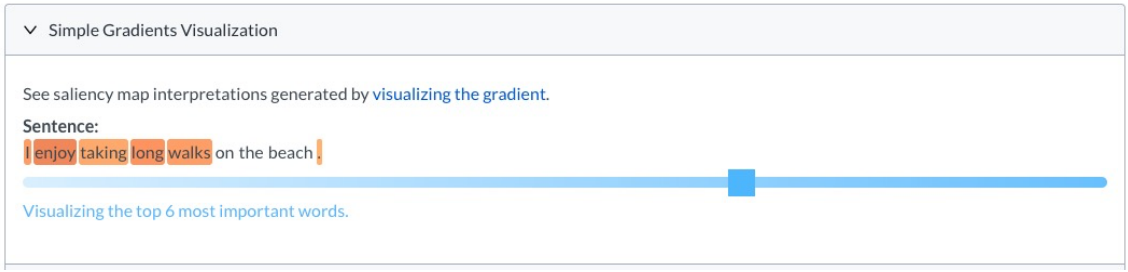


Figure 2.4: A saliency map visualising token importance for a sentiment analysis model. The map ranks the six most impactful words based on their gradient magnitude, shown in shades from dark to light orange. Such visualisations can aid in identifying tokens that, when transformed, are likely to influence model predictions. Obtained from AllenNLP Interpret (Wallace et al., 2020b).

- *Perturbation-based ranking:* Tokens are systematically perturbed to measure their impact on the loss. Methods include token removal (Jin et al., 2020; Garg and Ramakrishnan, 2020; Li et al., 2019) and replacement with neutral tokens like the out-of-vocab, unknown or mask token, or zero vectors (Ren et al., 2019; Ebrahimi et al., 2018b; Li et al., 2020b). Zhan et al. (2022) extend this to phrase-level importance.
- *Temporal metrics:* Gao et al. (2018) propose metrics based on the document's prefix and suffix relative to each token, termed the *temporal head score* and *temporal tail score*, and that quantify the token's influence on the model's output.
- *Attention-based ranking:* Attention scores from the victim model's layers are used to rank token importance (Hsieh et al., 2019; Maheshwary et al., 2021b; Zeng and Xiong, 2021). This is applicable only to models with attention mechanisms.

Yoo et al. (2020) found similar effectiveness across these methods,[3] with gradient-based approaches being more computationally efficient.[4] Subsequent works have proposed additional efficiency improvements, such as using Locality Sensitive Hashing (Charikar, 2002; Maheshwary et al., 2021b) or predictive networks (Guo et al., 2022) to reduce the computational burden.

Beam search and greedy search Beam search is a widely-used optimisation technique, often employed in language model decoding, that maintains multiple promising sequences (or "beams") simultaneously. The number of sequences is determined by the beam width parameter. A larger beam width generally leads to better solutions but increases computational cost. When the beam width is set to one, the method simplifies to greedy search, which selects the best transformation at each step without considering future implications.

In the context of adversarial attacks, beam search has been employed in works like Ebrahimi et al. (2018b,a) and has been reported to achieve high success rates, particularly for shorter documents, while being relatively query-efficient (Yoo et al., 2020). Variants of beam search, such as those with added randomness, have also been explored (Zhu et al., 2022). On the other hand, greedy search, used in Michel et al. (2019); Wallace et al. (2020a), is

[3] They did not test the attention method.

[4] Systematic perturbation methods need many forward passes through the model, but gradient methods only need a single forwards and backward pass.

computationally less demanding but can get stuck in local minima and may not find the global optimum. Liu et al. (2022) introduce a method called local search, which offers a middle ground, and provide theoretical performance bounds for it.

Population-based algorithms Population-based algorithms have been widely used for generating adversarial examples. These algorithms typically evolve a set of candidate solutions over time, guided by a fitness function that evaluates their quality.

Genetic algorithms: Among the most prevalent are genetic algorithms, which have been adapted for text-based adversarial attacks by Alzantot et al. (2020); Wang et al. (2021a); Maheshwary et al. (2021a). The process begins by generating an initial population of S candidates by applying S different transformations to the original example. In each iteration, new candidates are produced through a series of steps: application of transforms, fitness score calculation based on the loss, parent selection, crossover, and mutation. Different works employ various crossover functions; for instance, Alzantot et al. (2020) randomly select words from parent documents, while Wang et al. (2021a) splice parent documents at a specific point.

Differential evolution: As an alternative to genetic algorithms, differential evolution operates on continuous representations and has been used by Boucher et al. (2022).

Particle swarm optimisation (PSO): PSO has also been applied in this context. The algorithm initialises a swarm of particles, each with a position and velocity. Particles update their positions based on the best-known positions, both locally and globally. They then update their positions based on these best-known positions, with some randomness and a bias towards either the local or global best, depending on the variant. Zang et al. (2020) used PSO, while Chen et al. (2023) explored a variant known as quantum-behaved particle swarm optimisation (QPSO). Yoo et al. (2020) noted that PSO is query-inefficient despite its effectiveness in finding good solutions.

Other Various other search methods have been used. These have included

- *Learning-based approaches:* These methods integrate machine learning algorithms directly into the optimisation process. For instance, Emmery et al. (2021) use neural networks to identify which tokens in the text are most susceptible to perturbation. Similarly, Maimon and Rokach (2022) use a reinforcement learning algorithm to choose synonym swaps.
- *Minimising perturbation size:* Some approaches focus on generating adversarial examples with small perturbations to the original text. Xu et al. (2022) introduce a rollback operation that iteratively removes unnecessary edits, thereby reducing the perturbation size. In a similar vein, Lee et al. (2022) use a Bayesian optimisation technique that is further refined using importance scores and a post-processing step that minimises the perturbation size.

2.1.6 Constraints

As mentioned above, adversarial attacks all operate under a set of constraints. A constraint in this context refers to a rule or set of rules that determine what is an acceptable adversarial example. These constraints generally ensure that the perturbed input, while

still being effective in misleading the classifier, remains as close as possible to the original input in terms of semantics, structure, and readability. For instance, constraints may dictate that the modified text should not deviate significantly in meaning from the original example (semantic drift), must retain readability for human observers, should adhere to grammatical norms, and maintain a certain level of similarity between the original and adversarial examples. These constraints preserve the linguistic integrity of the adversarial samples and ensure that the attacks are realistic and plausible within the bounds of the assumed threat model.

As constraints will control the quality and characteristics of the generated adversarial examples, setting weaker linguistic constraints will increase the success rate of attacks, as the adversary has more freedom to alter the input text. However, the resultant adversarial examples will have lower text quality, which in this context, means they may seem unnatural or easily identifiable as adversarial interventions. Conversely, stronger constraints will produce higher quality adversarial examples that more closely resemble legitimate inputs. Yet, these constraints can also make it more challenging to find successful perturbations, and will lower the attack's success rate. This trade-off is why attack success rate cannot be used to compare attack algorithms with different constraints, as the attack with weaker constraints will perform better (Yoo et al., 2020).

In the following subsections, we categorise and present the different constraints that have been used in the literature. Table 2.4 provides a top-level overview.

2.1.6.1 Readability

These constraints are intended to enforce that the adversarial example be as close to natural text as possible.

Grammar A simple constraint in this area is that the adversarial example be grammatical. The intention is that adversarial examples that are more grammatical will naturally be of higher quality. This constraint can be checked using grammar-checking software, such as LanguageTool (Naber et al., 2003), as done by Lei et al. (2022)). While this approach ensures that the adversarial example follows standard syntactical rules, it overlooks the possibility of intentional grammatical deviations that might still convey meaning effectively in informal contexts. This means that this constraint is not applicable to all attacks or threat models. It wouldn't make sense when creating adversarial examples on data from domains which are rarely grammatical, such as tweets or forum posts. In these cases, the requirement for perfect grammar may hinder the creation of adversarial examples that maintain the linguistic characteristics of their target domain. For example, the attack by Tan et al. (2020) is aimed to deliberately mimic non-native English speakers, which makes grammar-checking less relevant in their context. Here, a grammatically correct sentence, "*The cat sat on the mat*" could be transformed to "*Cat sat on mat*", which in this context is a legitimate adversarial example.

Fluency Attacks which value fluency wish to produce smooth, naturally flowing text that is easily comprehensible and linguistically sound. These attacks prefer fluent sentences, like "*The quick brown fox jumps over the lazy dog*", over non-fluent ones, such as "*Quick brown fox jump over lazy dog*".

Category	Description	Sample references
Readability	Apply spellchecker	Belinkov and Bisk (2018)
Readability	Limit character transformations to words of specific length	Belinkov and Bisk (2018)
Readability	Keep first and last letters of each word	Belinkov and Bisk (2018)
Readability	Require pass through grammar checker	Lei et al. (2022)
Semantic	Check with linguistic acceptability model	Morris et al. (2020b)
Semantic	Fluency via language model perplexity	Wang et al. (2021b)
Semantic	Max of one transformation per token	Cheng et al. (2020); Ebrahimi et al. (2018b)
Semantic	Max of k transformations per token	Wang et al. (2021a)
Semantic	No replacing stop words	Jin et al. (2020); Xu and Du (2020)
Semantic	Only replace specific word types (e.g. noun)	Tan et al. (2020); Zang et al. (2020)
Semantic	Disallow stop words as replacements	Li et al. (2020b)
Semantic	Same part-of-speech tag for replacements	Jin et al. (2020); Zang et al. (2020); Xu and Du (2020)
Semantic	Filter replacements by fluency constraint	Alzantot et al. (2020); Gaiński and Bałazy (2023); Wang et al. (2021b)
Semantic	Attack only sentences with homographs	Emelin et al. (2020)
Semantic	Score replacements with class-conditioned language model	Lei et al. (2022)
Semantic	Min cosine similarity between document encodings	Jin et al. (2020); Garg and Ramakrishnan (2020); Cer et al. (2018); Liu et al. (2023)
Semantic	Forbid replacement of sentiment or negation words	Zhang et al. (2019)
Semantic	Min similarity on round-trip translation	Zhang et al. (2021)
Semantic	Max edit distance	Ebrahimi et al. (2018a); Gao et al. (2018); Li et al. (2019)
Performance	Include padding and unknown tokens in synonym set	Xu and Du (2020)
Performance	Use lemmatisation and delemmatisation for candidate synonyms	Zang et al. (2020)
Performance	Allow character transformations only for out-of-vocab words	Ebrahimi et al. (2018b)
Performance	Prioritize word replacements based on class and frequency	Samanta and Mehta (2017); Ren et al. (2019)

Table 2.4: Summary of the different constraints used in adversarial attacks.

The most common approach to measure fluency is to score the text by computing its likelihood according to a pre-trained language model, with the intention that more fluent text

will score higher than less fluent text. Scoring methods range from the simple measures of perplexity and entropy, to developed metrics like BARTScore (Yuan et al., 2021b) or BERTScore (Zhang* et al., 2020). These methods, akin to using one generative model to evaluate another (Deutsch et al., 2022), are widely used, often as a heuristic to select token replacements (Alzantot et al., 2020; Han et al., 2020; Gaiński and Bałazy, 2023). They can to overlook the potential loss of semantic meaning when making fluent substitutions. They may also not fully capture nuances like sentence complexity or more subtle aspects of fluency, such as tone.

An alternate approach is to assess the 'linguistic acceptability' of the text (Morris et al., 2020b). Linguistic acceptability is the task of determining if a sentence is acceptable or not, as judged by native speakers (Warstadt et al., 2018). This is evaluated using a model that has been trained on a linguistic acceptability dataset, such as the Corpus of Linguistic Acceptability (Warstadt et al., 2018), and that then assesses text as either acceptable or unacceptable. Fluent text is more likely to be marked as acceptable because of its preservation of natural grammatical structures. As an example from this dataset, "*The dog barked its way out of the room.*" would be acceptable, but "*The dog barked out of the room.*" would be unacceptable. This approach depends on the strength of the linguistic acceptability model, and often the model may fail to recognize degrees of fluency. Furtheremore, the model cannot judge cases where a sentence may still be acceptable but less optimal.

Character replacement constraints Certain constraints are specifically tailored for character-level attacks — the attacks that add, remove or replace individual characters to achieve success. These attacks often introduce spelling mistakes and typos. Hence, a constraint is to require that the adversarial example contain no spelling errors, according to a spellchecker. To improve readability, Belinkov and Bisk (2018) have enforced additional constraints on what character transformations are acceptable. They disallow character replacements for words less than a certain length, and require that the first and last letters of each word are preserved. The limitations of such methods include their inability to capture semantic shifts, as they focus solely on orthographic features. Examples of successful modifications under all these criteria might be changing *bead* to *bed*, or *house* to *horse*.

2.1.6.2 Semantic

Semantic constraints aim to ensure that the adversarial example preserves the original document's intent and meaning. These constraints are essential because in text, a single token change can significantly change the document's meaning. Consider the sentence: "*The doctor recommended regular exercise to improve health.*" Here, the word "*exercise*" is a key component of the sentence. If this word is changed to "*medication*" the sentence becomes "*The doctor recommended regular medication to improve health*", and the entire meaning of the sentence has shifted dramatically.

We can divide semantic constraints into a number of categories:

Token selection rules Constraints in this category govern how tokens in the original document can be transformed.

One type of constraint specifies which tokens can be acted upon, aiming to avoid changing tokens that are vital for document structure. For instance, some methods forbid the replacement of stop words (Jin et al., 2020; Xu and Du, 2020), or using stop words as a replacement Li et al. (2020b). Stop words are commonly used words in a language that contribute to the grammatical structure but carry little semantic weight, such as "*the*", "*is*", "at", and so on. For example, modifying stop words in a sentence like "*She is reading a book*" to "*She of reading a book*" significantly degrades the sentence's quality and changes its interpretation. A version of this constraint only specifically allows the replacement of specific word types, such as nouns, verbs, adverbs, or adjectives(Tan et al., 2020; Zang et al., 2020). While this ensures that only content-rich words are targeted, it overlooks the potential to subtly manipulate the meaning of a sentence by modifying less prominent but contextually significant words.

Another common constraint is to allow only one transformation per token (Cheng et al., 2020; Ebrahimi et al., 2018b), or more generally, up to k transformations per token (Wang et al., 2021a). This constraint aims to prevent preventing semantic drift caused by multiple token changes. An example, in the sentence "*The quick brown fox jumps over the lazy dog*", say the attacker replaces the second token, *quick*, with a synonym four times. The token changes from *quick* to *swift* to *immediate* to *instantaneous*, and the adversarial example "*The instantaneous brown fox jumps over the lazy dog*" has lost the meaning of the original. On the flipside, a limitation is that it can be too restrictive in situations where a series of smaller changes might be necessary to achieve a more meaningful attack.

Some token selection rules are specific to a particular NLP task. One example is for sentiment analysis attacks, where constraints may forbid the replacement of sentiment-expressing tokens to avoid changing the ground-truth label of the sentence. For example, if the original sentence is '*I am extremely happy with the service*', the constraints would prevent changing the sentiment-expressing word "happy". Modifying negation words like "not" could also be forbidden. These constraints are seen in Zhang et al. (2019). This constraint is focused on maintaining the sentiment of the text to ensure the attack does not inadvertently alter the task-specific outcome. But it will limit the scope of adversarial manipulations, and prevent the adversary from making subtle changes in sentiment-expressing words that might well go unnoticed in the overall attack strategy.

Token replacement rules Constraints in this category aim to ensure that, when replacing one token with another, the replacement is similar to the original in the context of the document.

One approach requires that replacements share the same part-of-speech tag as the original token, as seen in approaches by Jin et al. (2020); Zang et al. (2020); Xu and Du (2020). This ensures syntactic consistency but may overlook cases where a semantically similar word with a different part of speech could still work in context. Consider the sentence "*She was absolutely ecstatic about her results.*" The constraint would allow the adversary to change *ecstatic* (adjective) to *elated* (adjective), forming "*She was absolutely elated about her results.*", but would prohibit changing *ecstatic* (adjective) to *ecstasy* (noun), and then forming "*She was absolutely ecstasy about her results.*"

Another approach by Lei et al. (2022) attempts to maintain label preservation of the original class by evaluating replacements using a class-conditioned language model (Malmi et al., 2020). This is a pre-trained language model that is then fine-tuned on only exam-

ples of a particular class of the dataset, such as positive examples for a sentiment analysis dataset. If, after a token replacement, a text has a high likelihood according to one of these models trained on the class of the original example, the reasoning is that it must belong to a similar distribution as the training data — and hence, this text should be truly labelled as of original class.

Document embeddings Document embeddings represent entire documents as fixed-size, dense numerical vectors. They allow the rich and complex information contained in a document into a format that is easily interpretable by machine learning models. An earlier way to create them was to aggregating individual word embeddings, such as those from GloVe (Pennington et al., 2014). A more modern method would be to use a transformer, (e.g. T5 (Raffel et al., 2020)) to process the document and then pool the output embeddings. Adversarial attacks have tended to use the Universal Sentence Encoder (Cer et al., 2018) (USE), a model designed to produce sentence embeddings; examples include Liu et al. (2023) or Jin et al. (2020). A similar model used for other NLP tasks is the Sentence-BERT model (Reimers and Gurevych, 2019), which modifies the BERT architecture to produce embeddings more suitable to semantic similarity tasks.

Document embeddings operate on the principle that documents with similar meanings should have closely related embeddings. To evaluate the similarity between the original document and its adversarial counterpart, their embeddings can be compared. This comparison is often done using cosine similarity. If the similarity score is too low (below a certain threshold), it indicates a significant deviation in meaning, and as a result, the proposed transformation of the text is either avoided (Jin et al., 2020) or given a lower priority (Garg and Ramakrishnan, 2020).

Document embeddings operate on the principle that documents with similar meanings should have closely related embeddings. To evaluate the similarity between the original document and its adversarial counterpart, their embeddings can be compared. This comparison is often done using cosine similarity. If the similarity score is too low (below a certain threshold), it indicates a significant deviation in meaning, and as a result, the proposed transformation of the text is either avoided (Jin et al., 2020) or given a lower priority (Garg and Ramakrishnan, 2020).

Distance limits In many adversarial attacks on images (e.g. (Goodfellow et al., 2015)) it is common to put a constraint on the maximum distance between the original and adversarial example, typically measured with a L_p norm. This constraint is intended to ensure that the adversarial changes are imperceptible to the human eye, and the image appears unaltered. This concept of limiting the adversarial distance has been similarly applied in text-based adversarial attacks. As imperceptible changes are not possible, perhaps the equivalent goal here is to keep the adversarial text semantically and structurally close to the original, so that the changes are not so obvious to humans.

A common distance constraint is a maximum edit distance limit. Edit distance measures the number of transformations needed to change one text to another. One variant is the Hamming distance, which measures the minimum number of token replacements needed to convert between the original and adversarial example. Another is the Levenshtein distance, which also allows inserting and deleting tokens, on top of replacement. For instance, consider again the sentence "*The quick brown fox jumps over the lazy dog.*" If

we change *quick* to *swift* and *lazy* to *slow* the Hamming distance would be two, for the two token replacements. On the other hand, if we change *quick* to *swift*, remove *lazy* and insert *sleepy* before *dog*, the Levenshtein distance would be three, for one replacement, one deletion, and one insertion. Edit distance is commonly used (e.g. Ebrahimi et al., 2018a; Gao et al., 2018; Li et al., 2019)

Other distance metrics could include n-gram based metrics: those that track the number of matching n-grams between the original and adversarial examples. One example of this type of metric is BLEU (Papineni et al., 2002), commonly used in machine translation. Another proposed option is Word Mover Distance Zhang et al. (2020), which broadly speaking, tracks the minimum distance that words from one text need to 'travel' in embedding space to match exactly with the words in another document. However, these metrics have not been widely adopted.

2.1.6.3 Performance

Some constraints are intended to improve the attack's success rate, rather than to necessarily increase the text quality. Below are some examples of these.

Some approaches attempt to exploit the special tokens of the tokeniser to fool the victim model. For example, Xu and Du (2020) include padding and unknown tokens in the synonym set of words to increase the variety of potential replacements. This allows adversarial examples like "The [PAD] brown fox jumps over the lazy dog". Another approach along these lines is to allow character-level transformations only if the resulting word is not in the model's vocabulary, as done by Ebrahimi et al. (2018b). This is done so that victim models that tokenise on the word level will then represent the changed word with an out-of-vocabulary token, rather than with a different, valid word token. This can lead to more effective adversarial examples, as the model cannot leverage its understanding of the altered word. For example, if the sentence was "*I cannot sit down on this chair.*", the adversary could change *chair* to *chbir*, but not to *chain*. Then, *chbir* would be replaced at tokenisation with the out-of-vocabulary token. This method will be less effective for models that handle subword tokenisation.

To improve the search space of the attack, Zang et al. (2020) uses lemmatization when searching for synonyms, and delemmatization afterwards to prevent grammatical mistakes. Lemmatisation reduces words to their base or root form (lemma), while delemmatisation reverts lemmas back to their original inflected forms. For example, consider the sentence "*The cats are sleeping on the mat.*" With lemmatisation, *cats* would be reduced to its base form *cat* and similarly, *sleeping* to *sleep*. Then synonyms are searched for at the lemma level, which allows for a broader set of synonyms. After synonyms are found, like *feline* for *cat* and *rest* for *sleep*, delemmatisation is applied to fit these synonyms back into the original sentence context. The final sentence would then be "*The felines are resting on the mat.*" This approach expands the search space by enabling synonym substitution at the lemma level, but it may lead to unnatural word choices if the reversion process doesn't align perfectly with the context.

To increase attack strength, some attacks constrain replacements to those commonly seen in examples of other classes. For example, Ren et al. (2019) replace named entities with the most frequent same-type named entities from a different class. Named entities are specific, identifiable elements in text, such as names of people or places, that are cate-

gorised as distinct types through the task of named entity recognition. In this approach, named entities in a text are replaced with other named entities of the same type but often seen with examples of a different class to the original in the training dataset. For example, consider the sentence, "*I bought a smartphone from Apple.*" Here, *Apple* is a named entity, categorised as an organisation. With this strategy, *Apple* might be replaced with another organisation, such as *Samsung*, if *Samsung* is identified as a frequent entity for a different class. Similarly, Samanta and Mehta (2017) categorise each example (e.g., by movie genre for a dataset of movie reviews) and then prioritise word replacements that are frequently used both within that category and in examples from different classes.

2.1.7 Conclusion

In this survey, we have provided an overview of the key components that make up token-modification adversarial attacks: goal functions, transformations, search methods, and constraints.

We have first covered *goal functions*, which define what the adversary would like to achieve, and allow for a quantifiable measure of progress towards that goal. We have classified these as either classification or seq2seq goal functions, and again as either targeted or untargeted attacks, depending on the NLP task and the specifics of the attack goal. The majority of adversarial attacks have been for text classification, and it would be interesting to see more attacks targeting the wide range of seq2seq tasks, such as dialogue systems, paraphrasing, text summarisation and code generation.

We have then covered the wide range of proposed *transformations*: functions that alter one set of ordered tokens into another, where the token can be at a range of different granularities. Typical transformations insert, replace or delete tokens, with most attacks replacing and fewer inserting or deleting them. The type of transformation depends on the intent of the adversary. If they simply wish to induce a label flip for a classifier, they can change tokens as they like. If they wish to mimic human errors, they might introduce typos or phonetically similar words to the text. If they wish to create close-to-imperceptible adversarial examples, they may replace with visually similar tokens or use invisible characters. If they wish to preserve the fluency of the text, they might use a language model to mask a word, and then replace it — and so on.

After, we covered *search methods*. Starting from an original example, the search method repeatedly applies transformations until the adversarial example is found, or the search fails. They judge and choose transformations based on scoring functions, which numerically rank the available options. Search methods may be allowed access to the gradients of the victim model (white-box) or they may forbid access (black-box). Should the victim model not be available, search methods may use a substitute model instead. A variety of search methods have been proposed. Simple methods and heuristics, such as greedy search, run quickly but give worse results. More complex methods, such as particle swarm optimisation, give better results but are slower to run. Generally, there is a correlation between the number of times the victim model is queried, and the success of the search method; Yoo et al. (2020) has provided an initial benchmark of the performance and query effectiveness of search methods. More efficient search methods are continuing to be identified (e.g. Lee et al., 2022; Liu et al., 2022; Guo et al., 2022).

We have also surveyed the *constraints* of the field. Constraints determine what makes

successful or unsuccessful adversarial examples. They are used to enforce minimum standards on readability, fluency, grammatically and semantic similarity, or also to set a maximum allowed perturbation distance. Constraints can also increase the attack strength, such as to exclude transformations less likely to degrade the performance of the victim model. Constraints are often quantified using automated metrics, like the count of grammatical errors or language model entropy for assessing fluency. These constraints are generally employed at each stage of the optimisation algorithm to prevent the search process from deviating.

Finally, we will now propose some future directions that are motivated by the limitations of the field.

Standardise attack scenarios Currently the field is limited by a lack of standardised attack scenarios. The first problem is that each study establishes its own threat model, transformations, and constraints, which does not easily allow for cross-study comparisons. The second problem is that most papers' threat models do not align with real-world attack scenarios, which limits their practical relevance (Gilmer et al., 2018). Real-world scenarios might include, among others: circumventing email spam filters; manipulating user reviews to deceive sentiment analysis models; causing errors in translation models; altering news articles to evade fact-checking models; challenging hate-speech, offensive-language, or other automated content moderation systems; altering documents to evade piracy detection; or prompting chatbots to provide incorrect advice.

Extend to multimodal scenarios This has been a survey of text attacks, but it is worth remembering that many real-world systems are multimodal, integrating text, images, videos, audio, graphs, spatial data, structured data, time-series data, and more. For instance, webpages and emails often contain a mix of text, hyperlinks, images, video, and audio. Navigation systems meld text with images and geospatial data. A Facebook account may include a profile bio, a feed of text, video, and image posts, a network of friends, and geographical details (e.g., country of origin). Adapting text adversarial attack techniques to multimodal systems is a promising direction for future research.

More multiple-token attacks Most attacks insert, delete or replace single tokens at each step, which significantly limits the range of possible adversarial examples. By contrast, in Section 2.1.4.6 we discussed phrase-level attacks, such as the attacks by Xu et al. (2022) and Lei et al. (2022) that alter multiple tokens per optimisation step. These attacks are capable of a much larger range of adversarial examples, which is appealing, but to date they are much less explored. A promising option could be to incorporate generative seq2seq models, such as paraphrase models, to identify high-quality phrase and sentence-level transforms. Such an attack could modify phrases or idioms, which single-token attacks could not do. Consider the idiom "*Bite the bullet*", which refers to enduring a painful or unpleasant situation. Such an attack might replace it with a similar idiom, such as "*Grin and bear it*" or "*Tough it out*", which is not easily achievable with single-token modification algorithms.

Incorporate human judgements Currently attacks are often judged and ranked by automated metrics, such as fluency or semantic similarity scores. These metrics are flawed,

however, and often correlate poorly with human judgement (Deutsch et al., 2022; Morris et al., 2020a), leading to scenarios where many so-called 'adversarial' examples are non-fluent, ungrammatical, do not preserve semantics, change the ground-truth label, etc. (Herel et al., 2022). The core problem is that it is very hard to capture all the required 'adversarial' properties with a collection of metrics, as the ultimate judge of adversarial attacks is a human, and the metrics deviate from human judgement. Directly incorporating human feedback would improve the quality of adversarial examples, but human judgements are typically difficult and expensive to obtain.

Two alternatives come to mind. The first is to allow the end user to construct the adversarial example, perhaps with guidance from a user-friendly interface and assistance from a model, as demonstrated in (Wallace et al., 2019b). The second is to use techniques like Reinforcement Learning from Human Feedback (RLHF). Here, an adversary would develop a 'reward' model that assigns scalar rewards to assess the quality of adversarial examples. The reward model would be trained using human preferences, and would be applied as a constraint in the optimisation process. The use of RLHF was key to the success of ChatGPT and other similar systems. It is particularly useful for scenarios where defining an explicit reward function is difficult, such as for this one.

2.2 Generative Adversarial Attack Preliminaries

This section makes up the second part of the literature review and sets the groundwork for the remaining research questions of the thesis. The focus is on generative approaches for textual adversarial examples. We will give a brief overview of transformers, including the pre-train and fine-tune paradigm and the different decoding methods. We will introduce the existing literature on generative adversarial attacks and discuss the training objectives required to train these models. Finally, we will tackle some specific topics useful for our research questions: namely, policy gradients for Chapter 3 and multilingual models for Chapter 5.

2.2.1 Transformers

Prior to 2017, the most popular models for NLP tasks were Recurrent Neural Networks (RNNs) and their variants, such as the Long Short-Term Memory (LSTM) network. These model architectures process data sequentially, in that each decoding step only uses information from the previous decoding state, which is passed in the form of a hidden state vector. These models had two main drawbacks. The first was that their sequential nature made it difficult to parallelise batches of sentences. The second was that they struggled to capture long-range dependencies in the text, as they relied only on the hidden state vector.

The transformer model, introduced by Vaswani et al. (2017), has mitigated both of these problems. Transformers process text not sequentially, but as a whole, meaning that batches of sentences can be parallelised. They also include self-attention layers that allow the context of the entire text to be considered, which eliminates the sequential dependency problem and effectively captures long-range dependencies. Although transformers have their weaknesses, such as the substantial memory requirements for large models and the intrinsic sequential nature of the decoding process, over the years transformers and its variants have become the most popular architecture for a wide range of NLP applications,

consistently setting new performance benchmarks. They are currently the go-to model of the field.

Transformers may possess either an encoder, a decoder, or both of them. An encoder processes the input sequence and compresses the information into a numerical vector that contains a distilled representation of the input. Encoders are used for tasks that require an understanding of the entire input sequence, such as sentence classification or named entity recognition. Decoders, on the other hand, are used to generate an output sequence from a numerical vector, and are used for text generation tasks. When encoders and decoders are used together, they are referred to as an encoder-decoder architecture, and are suitable for seq2seq tasks like translation, summarisation, or the generation of adversarial examples.

The first step in the NLP pipeline for transformers (and for other NLP models) is tokenisation, where the raw text is converted into a numerical form that can be processed by a machine learning model. During tokenisation, the input text is converted into a series of tokens using a tokenisation algorithm. The tokens can be of different levels, but the the most common are character, word, and subword. In character-based tokenisation, each character is a token, which results in a small overall vocabulary but a large number of tokens per input text. Word-based tokenisers segment text into words, which leads to a large vocabulary that can become cumbersome. Subword tokenisers allow for a middle ground and have become commonly used. Special tokens (like [CLS], [SEP], [BOS], [EOS], and [UNK]) are also used to provide more helpful context to the model. They can be used to signify the start and end of sequences, identify unknown tokens, denote special events, and so on. WordPiece (Wu et al., 2016) is a common tokenisation algorithm used by BERT (Devlin et al., 2019) and many BERT variants, such as DistilBERT (Sanh et al., 2019), Electra (Clark et al., 2020) and MobileBERT (Sun et al., 2020). SentencePiece (Kudo and Richardson, 2018) is another common tokenisation algorithm, and is used by the transformer architectures T5 (Raffel et al., 2020), XLNET (Yang et al., 2019) and LLAMA (Touvron et al., 2023), amongst others.

After tokenisation the tokens are converted to embeddings, which are compact numerical vectors that represent the semantic information of the input. The embeddings are then passed through a series of self-attention and feed-forward layers, and after multiple layers of such processing, the final output embeddings are generated. For decoders, the output embeddings are then used to generate the text sequence, which we cover in Section 2.2.1.2.

Many variants on transformers have been proposed and embraced. Vaswani et al. (2017) introduced the transformer architecture. It was extended by the GPT model (Radford et al., 2018), which was the first transformer to do unsupervised pre-training, and then fine-tuning on a target task. Later, its successor GPT-2 (Radford et al., 2019) expanded the architecture and scale. BERT (Devlin et al., 2019) introduced a bidirectional attention mechanism to produce better sentence embeddings, and later came DistilBERT (Sanh et al., 2019), a lighter and faster distilled variant with close-to-equivalent performance. T5 (Raffel et al., 2020) and BART (Lewis et al., 2020) returned to the original encoder-decoder structure but with different pre-training objectives. XLNet (Yang et al., 2019) tried to combine the best of GPT and BERT with a permutation-based training strategy. Longformer (Beltagy et al., 2020) addressed the limitation of sequence length by introducing sliding-window attention. Many more architectures and variants have been proposed. Recently, transformers also have been adapted to tackle non-NLP tasks, with examples

including computer vision (Khan et al., 2022), time series forecasting (Wen et al., 2023), protein structure prediction (Lin et al., 2023) and many others.

Lately, the trend has been to increase the size and complexity of models in order to achieve better performance, as illustrated by models like GPT-3 (Brown et al., 2020). However, these larger models also have high computational and memory requirements. Hence, an active area of research has been in model distillation (e.g. for DistilBERT (Sanh et al., 2019)), which attempts to compress the knowledge of a large model into a smaller one while achieving comparable performance. Distilled models are very useful for edge deployments, latency-sensitive applications, or more generally, when GPU memory or computing resources are limited. We have made heavy use of distilled models through our research in Chapter 3 to 5 to make the most of our computing environment which has limited GPU memory.

2.2.1.1 Pre-training and fine-tuning

The pre-train and fine-tune paradigm has gained widespread acceptance in the NLP community, primarily because it allows models to generalise well across various tasks with minimal task-specific annotations. In the pre-training phase, the model is typically trained on a unsupervised learning task (such as to predict masked tokens in a sentence) over a very large corpus. As the task is unsupervised, labelled data is not required, which allows for use of a very large corpus — such as the Common Crawl, a corpus of web crawl data consisting of over 50 billion webpages. In this pre-training stage, the model learns a useful representation of the language structure, semantics, and world knowledge that is implicitly contained in the corpus and is useful across many NLP tasks.

In the fine-tuning phase, the pre-trained model is then trained again, this time directly on the NLP task of interest. A dataset specific to the task is used, and the fine-tuning training objectives are also task-specific loss functions, such as cross-entropy loss for classification tasks. By starting from a good representation and then fine-tuning, the model will require less training data, converge faster, and will (tend to) achieve superior performance when compared to a model directly trained only on the task-specific dataset.

As pre-training models is expensive and requires a lot of training data, usually only large corporations or research labs will pre-train a model from scratch. More common is to make use of a variety of previously pre-trained models that have been shared freely with the community, easily accessed through services like the Hugging Face Model Hub.[5] We have used many of these pre-trained models in our research.

2.2.1.2 Decoding methods

In the decoding phase, transformers generate text one token at a time, from left to right in an auto-regressive manner. The probability of generating a specific token w_t is conditional on the prior generated tokens of the sequence, and is indicated by $P(w_t|w_{1:t-1})$. The probability of generating a given sequence with T tokens can then be calculated as

$$P(w_1, w_2, ..., w_T) = \prod_{t=1}^{T} P(w_t|w_{1:t-1}). \tag{2.1}$$

[5] https://huggingface.co/models

A number of different decoding strategies exist, each making use of this conditional distribution, and each with its own strengths and weaknesses.

The simplest decoding strategy is *greedy decoding*. This is where at each step the next chosen token is the one with the highest conditional probability, so that at timestep t, $w_t = \arg\max P(w_t|w_{1:t-1})$. Greedy decoding is straightforward and efficient, but tends to lead to repetitive or low-quality text. An alternate strategy, *beam search*, keeps track of the top-k most probable sequences (beams) at each time step. The sequences are scored based on their log-likelihood, and only the top-k sequences are retained. The number of beams k is a parameter that affects the trade-off between quality and computational cost. The method is more likely than greedy search to find a globally optimal sequence, but can be computationally intensive, and the generated text sequences tend to be repetitive, generic, and lack diversity (Vijayakumar et al., 2018).

Human language tends not to select most probable tokens, but is instead surprising, selecting sometimes low-probability tokens (Holtzman et al., 2020). This has inspired the *sampling* decoding method, where at each time step a token is randomly picked from the next-token distribution, $w_t \sim P(w|w_{1:t-1})$. This makes the text more diverse, but also tends to make it lose coherence and often not make sense (Holtzman et al., 2020), but this can be mitigated by adjusting the *temperature* parameter of the conditional next-token distribution. The temperature parameter refers to τ when the softmax is calculated for each token in the vocabulary in the equation $\frac{\exp(w_t/\tau)}{\sum_{j \in V} \exp(w_j/\tau)}$. The temperature controls the entropy of the distribution, where higher temperatures result in a more uniform distribution that leads to more randomness in the samples, and lower temperatures result in a more deterministic distribution, and a sampling distribution closer to greedy decoding.

The final decoding method we will mention here is *diverse beam search* (Vijayakumar et al., 2018), a beam search variant that aims to increase the diversity between generated text samples. In this approach, beams are partitioned into g groups, where each group maintains its own set of top-k beams. Within each group, a controllable diversity penalty is added to the score of each candidate token based on its frequency of occurrence in the same group. The idea is to discourage the selection of the same token across multiple beams within a group. This method balances exploration and exploitation: it explores different thematic directions by introducing diversity, while still exploiting the high-probability sequences to generate coherent and contextually relevant text. We have found this method particularly useful for generating adversarial examples and we use it heavily when addressing our research questions (Chapter 3 to Chapter 5).

Finally, in combination with the other decoding methods, we have used a trick commonly used to increase coherence called *top-p sampling*, where at each step the decoder only samples from a subset of tokens that accounts for the majority of the probability mass. The subset is chosen by sorting tokens by their conditional probability $P(w_t|w_{<t})$, and then selecting the smallest set whose cumulative probability exceeds a predefined threshold p. The parameter p typically lies between 0.9 and 1 and controls the trade-off between randomness and faithfulness to the original distribution.

2.2.2 Generative Adversarial Attacks

Generative models, which can produce text conditioned on various prompts or contextual information, have been successfully used to generate adversarial examples for a wide

range of NLP tasks. The most popular is for text classification, but generative models have also been trained to produce adversarial examples for question answering Lu et al. (2022), to generate misleading explanations for NLI models (Camburu et al., 2020), dialogue systems (Li et al., 2017), to deceive fake news detection algorithms (Le et al., 2020), and so on.

A wide range of generative model types have been used for generating adversarial examples. Feed-forward networks have been used to generate answers in multiple-choice visual question answering tasks, where they create distractors to mislead victim models (Lu et al., 2022). Simple matrices that learn example-dependent token probabilities have also been used to generating adversarial examples (Guo et al., 2021). RNNs and LSTMs have been used to paraphrase sentences, where they transform text according to specific parse templates (Iyyer et al., 2018; Vijayaraghavan and Roy, 2019; Camburu et al., 2020). Generative Adversarial Networks (GANs) have been explored but are often considered challenging to train for this task due to issues like mode degeneracy and semantic divergence (Zhao et al., 2018; Wong, 2017). Autoencoders and variational autoencoders (VAEs) have also been used (Xu et al., 2021; Ren et al., 2020). There are only a few examples of pre-trained transformers being fine-tuned for adversarial generation (Fursov et al., 2021; Qi et al., 2021), but they have showed promise for generating high-quality adversarial examples efficiently.

When training a model to generate text in a supervised fashion (i.e. when ground truth labels are available), the training data will consist of pairs of input-output sequences. By contrast, adversarial example generation is a self-supervised task, where no ground-truth output is available and the input is used to guide the task. The process typically involves training the generative model with an objective function that incorporates a variety of components. The most important component is an adversarial component, which provides a measure of the victim model degradation for the generated text when compared to the original. Other components aim to maintain text quality, such as grammar or semantic similarity metrics, or others with a range of purposes, such as diversity components, KL divergence components, architecture-specific components, task-specific components, and so on. The components are typically added together with different weights to form the objective function, and during the optimisation process, the optimiser adjusts the parameters of the model to minimise this objective function. The components are typically in conflict with each other: for example, focusing excessively on adversarial loss will generate text that lacks grammatical fluency. Finding the optimal balance between components is difficult, as placing undue emphasis on any component will lead to sub-optimal results.

In a white-box adversarial setting, where gradient information from the victim classifier is available, efficient gradient-based optimisation algorithms like SGD and Adam can be used. These methods rely on gradients to iteratively update the generative model in a direction that minimises the loss function, enabling rapid and precise convergence. They can also allow for some interesting model designs, such as in Wang et al. (2020b), which incorporate a downstream model which allows the generative model to control the topic of its generated adversarial candidates at inference time. In contrast, in a black-box adversarial setting, where there is no gradient access, alternative optimisation techniques are used. These include using substitute models, policy gradients, or modelling without any knowledge of the victim model, as in (Zhao et al., 2018).

In each of our research works, we have presented a related work section that describes how

that particular work relates to the previously trained generative models of the literature. We refer the reader there for more details.

2.2.3 Policy Gradients

In Chapter 3, we have fine-tuned a transformer using a variant of REINFORCE, a reinforcement learning policy gradient algorithm. In this section we give a very brief overview of this particular policy gradient algorithm. There are many others, but the basic principles are consistent across them.

Under the reinforcement learning framework, we have a Markov process with an agent at some time t and state S_t. The agent takes an action A_t in some environment, for which it collects a reward R_{t+1} and transitions into a new state S_{t+1}. This gives a series $S_0, A_0, R_1, S_1, A_1, \ldots$ which is called the trajectory τ of the agent.

The agent chooses actions according to a probabilistic policy $\pi(A_t|S_t; \boldsymbol{\theta})$. This is a probability distribution function defined by a parameter vector, $\boldsymbol{\theta}$, and it is dependent on the state S_t. The agent aims to find the parameter vector $\boldsymbol{\theta}$ that results in the highest $r(\tau)$, the sum of rewards for a given trajectory τ. This leads us to the goal function for the agent:

$$J(\boldsymbol{\theta}) = \mathbb{E}_\pi[r(\tau)]$$

which incentives the agent to maximise the expected reward over all trajectories. The expectation when written out becomes

$$\mathbb{E}_\pi[r(\tau)] = \int \pi(\tau) r(\tau) d\tau$$

which is the sum of multiplying the probability of each trajectory τ with its total reward $r(\tau)$, while assuming the agent follows its policy π.

The usual approach to finding $\boldsymbol{\theta}$ is by gradient descent, in which the gradient $\nabla J(\boldsymbol{\theta})$ is used to make updates to $J(\boldsymbol{\theta})$. This gives the parameter update rule

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \nabla J(\boldsymbol{\theta})$$

where α is a step size parameter.

The goal is then to find $\nabla J(\boldsymbol{\theta})$, or in other words, to find $\nabla J(\boldsymbol{\theta}) = \frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \frac{\partial \mathbb{E}[r(\tau)]}{\partial \boldsymbol{\theta}} = \frac{\partial}{\partial \boldsymbol{\theta}} \int \pi(\tau) r(\tau) d\tau$. (We will use the ∇ notation from here on as it is easier). From here this simplifies to $\nabla E_\pi[r(\tau)] = E_\pi[r(\tau) \nabla \log \pi(\tau)]$. This is known as the **policy gradient theorem**.

To compute this, you will have to compute $\pi(\tau)$, the probability of a particular trajectory under the policy. This is given by $\pi(\tau) = p(s_0) \prod_{t=1}^{T} \pi(a_t|s_t) p(s_{t+1}, r_{t+1}|s_t, a_t)$: the probability $p(s_0)$ of the agent starting in s_0, multiplied by the chance of taking each defined action a_t at state s_t according to the policy, then the chance of getting to the next state s_{t+1} and getting the reward r_{t+1} from s_t, a_t. This latter probability is determined by the environment and not by anything the agent can control. In many environments it is difficult to calculate an exact value for this probability, but luckily the policy gradient

theorem can be simplified further to get $\nabla E_\pi[r(\tau)] = E_\pi\left[r(\tau)\left(\sum_{t=1}^{T}\nabla \log \pi(a_t|s_t)\right)\right]$. This particular formulation factors out the environment factors $p(s_{t+1}, r_{t+1}|s_t, a_t)$ and $p(s_0)$, which is good because now it is much easier to calculate.

The next problem is the expectation term. It is usually not feasible to calculate this since we would need to evaluate the total rewards for all possible trajectories in the environment, which is far too many to enumerate for most scenarios. One approach is to take a number of trajectory samples and use them as an estimate of the underlying process (in other words, a Markov Chain Monte Carlo estimate). This leads to the REINFORCE algorithm, which takes S samples and works out the gradient as $\nabla E_\pi[r(\tau)] = \sum_{s=1}^{S}\sum_{t=1}^{T} G_t \nabla \log \pi(a_t|s_t)$, where $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots$ is the discounted return: a weighted sum of the previous rewards at each time step (and a rough approximation of $r(\tau)$). This is an unbiased estimator with variance dependent on how many samples are taken. More samples reduce the variance, but are also more computationally expensive.

These are the basics of REINFORCE and policy gradient algorithms. Many variants have been developed, but they are out of scope for this section.

2.2.4 Multilingual Models

In Chapter 5 we have produced a generative adversarial attack against a multilingual victim model. Multilingual models are those that can handle a range of languages, typically trained using corpora that contain many languages. Once pre-trained, a multilingual model can be later fine-tuned in any language it has been pre-trained on, and can be used for the same NLP tasks as monolingual models. They are particularly good at innately multilingual tasks, such as machine translation or classification on code-switched text (text that contains more than one language, such as English and Hindi) (Tan and Joty, 2021).

Multilingual models do not learn representations of each language separately, but instead use the common elements across languages (e.g. word stems, conjugation patterns, grammar structures) to improve its general understanding across them. This has particular benefits for low-resource languages, where there would not be enough data to train a monolingual model on the language itself, and the model can generalise what it has learned to the low-resource language. The language balance in the pre-training data will affect the performance. Training a multilingual model on too many languages can lead to the problem of language interference (also called the *curse of multilinguality* (Pfeiffer et al., 2022)) where the interference between languages degrades the overall performance of the model.

In terms of architecture, multilingual models have shared embedding space across languages and a correspondingly larger vocabulary than monolingual models. The shared embedding allows the model to be of a reasonable size, while still performing well across languages. Some examples of multilingual models include mT5 (Xue et al., 2021), mBERT (Devlin et al., 2019), XLM (Lample and Conneau, 2019) and mBART (Liu et al., 2020).

Chapter 3

A Constraint-Enforcing Reward for Adversarial Attacks on Text Classifiers

In this chapter we answer the second research question outlined in Section 1.2.

Abstract. Text classifiers are vulnerable to adversarial examples — correctly-classified examples that are deliberately transformed to be misclassified while satisfying acceptability constraints. The conventional approach to finding adversarial examples is to define and solve a combinatorial optimisation problem over a space of allowable transformations. While effective, this approach is slow and limited by the choice of transformations. An alternate approach is to directly generate adversarial examples by fine-tuning a pre-trained language model, as is commonly done for other text-to-text tasks. This approach promises to be much quicker and more expressive, but is relatively unexplored. For this reason, in this work we train an encoder-decoder paraphrase model to generate a diverse range of adversarial examples. For training, we adopt a reinforcement learning algorithm and propose a constraint-enforcing reward that promotes the generation of valid adversarial examples. Experimental results over two text classification datasets show that our model has achieved a higher success rate than the original paraphrase model, and overall has proved more effective than other competitive attacks. Finally, we show how key design choices impact the generated examples and discuss the strengths and weaknesses of the proposed approach.

3.1 Introduction

Adversarial attacks cause a *victim model* — an attacked machine learning model — to make a specific mistake. These attacks occur across domains, pose a real-world security threat[1] and are increasingly well-studied (Biggio and Roli, 2018; Zhang et al., 2020). In this paper we study adversarial attacks on text classifiers, where an adversary takes a correctly-classified *original* example and perturbs it to create an incorrectly-classified *adversarial* example. The adversarial example must typically meet some acceptability constraints (e.g., a maximum edit distance from the original, preserving semantic meaning, grammaticality), although there is no general consensus on these (Morris et al., 2020b).

[1]For example, (Wallace et al., 2020a) attacked Google Translate with adversarial examples, causing vulgar outputs, word flips, and dropped sentences.

Elsewhere, the tendency is towards more developed packaging than before.
Elsewhere the tendency is to favour more developed packaging than the previous.
Elsewhere the trend is toward more developed packaging than before.
Elsewhere the trend is towards more developed packaging than before.

The net sales decreased to EUR 49.8 million from EUR 59.9 million.
Net sales were limited to EUR 49.8 million from EUR 59.
Net sales were limited to EUR 49.8 million from EUR 59.9 million.
The Net sales were limited to EUR 49.8 million from EUR 59.
The net sales were limited to EUR 49.8 million from EUR 59.
The net sales were limited to EUR 49.8 million from EUR 59.9.
net sales were limited to EUR 49.8 million from EUR 59.9 million.

Figure 3.1: Examples of successful adversarial attacks against a sentiment classifier obtained with the proposed approach. On top, the adversarial examples flip the sentiment from the original neutral (blue) to positive (green), and on bottom, sentiment goes from the original negative (red) to neutral (blue).

How are text adversarial examples found? The predominant approach is to repeatedly modify tokens until the predicted label changes (Zhang et al., 2020). Attacks taking this approach, known as token-modification attacks (Roth et al., 2021), find adversarial examples by solving a constrained combinatorial optimisation problem. First, they define the success condition, the constraints and the allowed transformations, and then they use a search algorithm to seek a solution (Morris et al., 2020a). While effective, these attacks also have major downsides. Firstly, they are slow: the computational budget heavily impacts their success rate, with high-performing search algorithms requiring many victim model queries per example, particularly for long texts (Yoo et al., 2020). Secondly, their allowed token-level transformations limit their search space, largely preventing complex transformations like paraphrasing or style change.

Alternatively, the adversarial example task can be formulated as a text-to-text problem, with original examples as input and adversarial examples as output. It could then be straightforwardly approached with seq2seq models, as done for other text-to-text tasks like summarisation or translation. This approach enjoys several principled advantages over token-modification attacks. Firstly, once trained, finding adversarial examples is much faster (in the order of a few milliseconds, rather than minutes or even hours). Additionally, through beam search or sampling, this approach can easily generate multiple adversarial examples per given input, while also controlling their diversity, tonality, or other characteristics. Finally, a seq2seq approach is also intrinsically more flexible as it is not limited by a rigid set of allowed transformations.

On the other hand, the main challenge of this approach is that it is notoriously difficult to train a model to generate controlled text. The training process can be unstable and time-consuming, and the generated text can be ungrammatical, irrelevant, nonsensical, unnatural, bland, repetitive, or incoherent (Holtzman et al., 2020; Hu et al., 2017; Wong, 2017). For our task there is an additional challenge: the generated text must change the victim model's predicted label while not violating any constraint.

For these reasons, in this paper we propose fine-tuning a pre-trained encoder-decoder paraphrase model so that it produces adversarial examples instead of paraphrases. We fine-tune using a reinforcement learning (RL) policy-gradient algorithm — REINFORCE

with baseline (Williams, 1992) — and attack a sentiment classifier. For training, we propose an original reward function that both incentivises adversarial examples and penalises any violation of the constraints. To improve generated text coherence, our loss function includes a Kullback-Leibler (KL) divergence term (Kullback and Leibler, 1951) that limits parameter drift from the pre-trained paraphrase model. The attack requires the victim model's prediction confidence, but no other information, which makes our attack either a *grey-box* (Biggio and Roli, 2018) or a *black-box* attack (Zhang et al., 2020), depending on the definition.[2]

We have evaluated the proposed approach on two sentiment analysis datasets, reporting the attack success rates and the diversity of the generated adversarial examples across four different decoding methods and two training temperatures. The results show that the the proposed approach has been able to generate numerous and diverse adversarial examples, with success rates much higher than for the pre-trained paraphraser and comparable token-modification attacks. In summary, this paper makes the following key contributions:

1. an approach for the generation of adversarial attacks to text classifiers based on a pre-trained paraphraser and reinforcement learning;
2. a constraint-enforcing reward function that incentivises adversarial examples and penalises constraint violations;
3. experimental results on two text classification datasets showing the effectiveness of the proposed approach, and a comprehensive analysis and discussion.

3.2 Related Work

For ease of reference, we can divide the literature on text classification adversarial attacks into token-modification attacks and generative attacks.

Token-modification attacks. The vast majority of existing text adversarial attacks are token-modification attacks. They consist of four main components: a goal function, a set of allowed transformations, a set of constraints that must be satisfied, and a search method (Morris et al., 2020a). These approaches typically create adversarial perturbations by applying repeated token-wise transformations, such as character replacements (Ebrahimi et al., 2018b) or synonym swaps (Ren et al., 2019)[3]. A detailed description of these attacks is not relevant to our work, so the reader can refer to a recent survey (Roth et al., 2021) for further details.

Generative attacks. Some previous work has attempted to train a variety of generative models to produce adversarial examples. For example, long short-term memory variants have been used by Iyyer et al. (2018) to paraphrase a sentence in the form of a parse template, and by Vijayaraghavan and Roy (2019) to perturb examples. A feed-forward network was used by Lu et al. (2022) to generate distracting answers in a multiple-choice

[2]These assumptions are not unrealistic: for example, most pre-trained models on the Hugging Face Model Hub report both predictions and confidences.

[3]These approaches typically produce one adversarial example per original. TextAttack (Morris et al., 2020b) — the most popular library for token-modification attacks — has been set up to only return at most one adversarial example per original, so this is what we have used throughout this paper. While more could be searched for, it has not been well studied how they could be found efficiently in incremental time.

visual question answering task. Other work has attempted to use GANs and autoencoders (Zhao et al., 2018; Ren et al., 2020; Wong, 2017). However, this line of approach has not been widely pursued, probably due to training difficulties. For example, Wong (2017) reported widespread issues like mode degeneracy, semantic divergence, and reward hacking. Furthermore, it is challenging to maintain certain text properties, such as label invariance, while generating text.

Since their introduction, transformers have become a ubiquitous encoder-decoder architecture in contemporary natural language processing. They are typically trained with transfer learning, first solving a large-scale pre-training task (typically unsupervised or self-supervised), and then fine-tuning on the target task. Large transformers such as T5 (Raffel et al., 2020) currently achieve state-of-the-art performance in many text-to-text tasks. Despite this success, no previous work we are aware of has attempted to fine-tune a pre-trained paraphrase model for adversarial example generation, as the proposed approach does. The closest works are Gan and Ng (2019), who create a dataset of adversarial paraphrases manually, and Qi et al. (2021), who use a pre-trained text style transfer model, but do not fine-tune it.

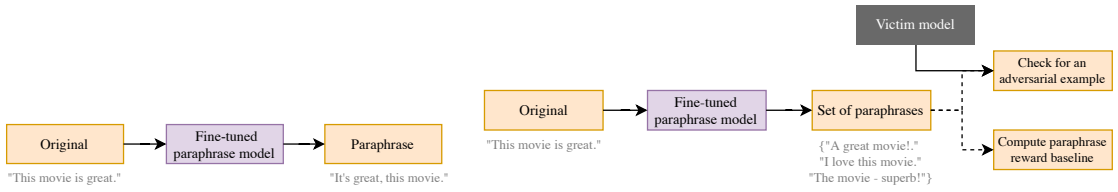
3.3 Proposed Approach

3.3.1 Overview

Our overall goal is to fine-tune a pre-trained paraphrase model with a reinforcement learning objective so that it can learn to generate adversarial examples. We use a T5 transformer as the base pre-trained model.

During each training epoch, we generate one paraphrase per original example and collate them into batches of training data. The batches are used to compute a loss function (Section 3.3.2), which incorporates both a reward function (Section 3.3.3) and a baseline (Section 3.3.4). We use a set of constraints (Section 3.3.5) to determine if the generated text is valid, and examples that fail receive zero reward. Figure 3.3 shows the overall setup.

During validation, we are able to generate not one, but a whole set of adversarial example candidates per original example by simply using off-the-shelf decoding (Section 3.4.2). We call these generated examples the set of *adversarial example candidates*, and consider the attack successful if at least one of them meets the given constraints. Since the limited resource are the available original samples, we compute the attack success rate as the ratio between the number of successful attacks and the number of original examples. Generating more paraphrases can obviously improve the attack success rate, but the generation takes longer and the memory requirements increase. As an effective trade-off between these factors, we have chosen to generate $n = 48$ paraphrases per original example. The same procedure is used at test time. During validation, we also update the reward baseline with the average per-example reward across the candidate set. Training is stopped once the validation set performance improvement drops below a threshold, or after a maximum number of epochs (full details are available in Section 3.9).



(a) (b)

Figure 3.2: Sample generation during training and validation. (a) During training, we generate one paraphrase per original example, decoding with nucleus sampling. (b) During validation, we generate a set of paraphrases per original example, decoding with one of four methods (Section 3.4.2). We then check if any paraphrase in the set is a successful adversarial example, and also use the set (for the training split) to update the reward baseline (Section 3.3.4).

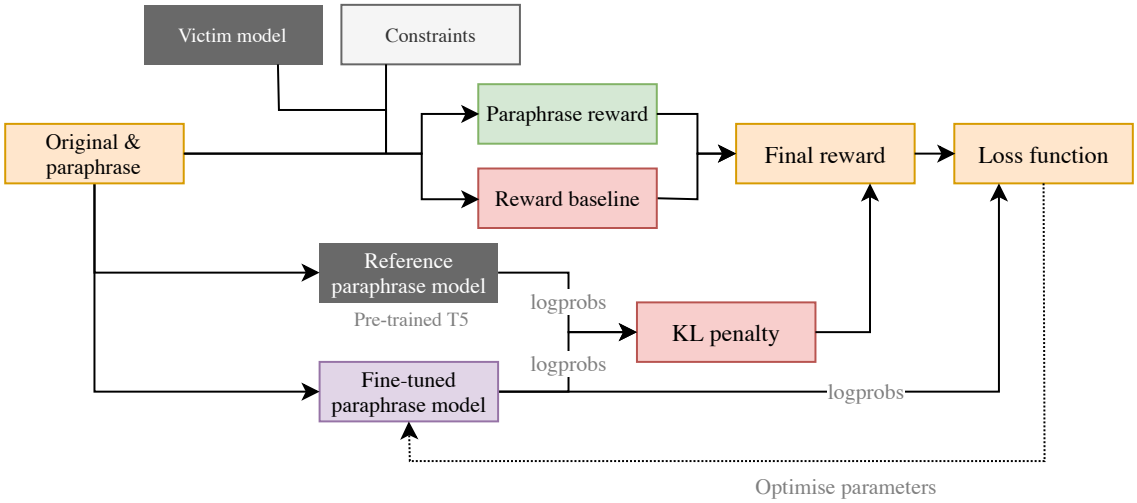


Figure 3.3: A diagram of the training approach. As input, training uses batches of *(original, paraphrase)* pairs. The parameters are updated using a REINFORCE with baseline algorithm. The overall loss function depends on the reward function, the baseline, the constraints, and the KL divergence penalty, which compares the probabilities computed by the fine-tuned and pre-trained paraphrase models.

3.3.2 Loss function

Let us have an input data distribution $\mathcal{D}$, an original example $x \in \mathcal{D}$, and a pre-trained paraphrase model with parameters θ. Given x, the model generates a paraphrase x' with T tokens with probability $p_\theta(x'_t|x'_1, \ldots, x'_{t-1}, x), t = 1 \ldots T$. We note this predictive distribution as ρ for simplicity. Our agent attempts to learn a policy π, still parametric in θ and initially equal to ρ, that can create adversarial examples, for which we have a reward function r that scores success and failure. Training aims to optimise π to maximise the expected value of r:

$$\mathbb{E}_\pi(r) = \mathbb{E}_{x\sim\mathcal{D},x'\sim\pi(x)} r(x, x')$$

To optimise using gradient descent, the gradient $\Delta_\theta \mathbb{E}_\pi(r)$ is required, for which an esti-

mator is provided by the policy gradient theorem (Sutton et al., 1999):

$$\Delta_\theta \mathbb{E}_\pi(r) = -r \sum_{t=1}^{T} \frac{\partial}{\partial \theta} \log \pi(x'_t | x'_1, \ldots, x'_{t-1}, x) \tag{3.1}$$

where x' is a sampled sequence (using any of a number of sampling methods). The above is the REINFORCE estimator (Williams, 1992). Using an automatic differentiation framework we can convert this into a loss function:

$$\begin{aligned} L_{RF} &= -r \sum_{t=1}^{T} \log \pi(x'_t | x'_1, \ldots, x'_{t-1}, x) \\ &= -r \log \pi(x'|x) \end{aligned} \tag{3.2}$$

This estimator is unbiased, but it typically exhibits a large variance, which causes slow and unstable learning. The variance can be reduced by subtracting a baseline, b, from r:

$$L_b = -(r - b) \log \pi(x'|x) \tag{3.3}$$

provided b is highly correlated with r. This estimator is biased in the case that b depends on x' (Williams, 1992), but typically delivers improved training speed and stability.

Concurrently, we also like to prevent the trained distribution, π, from diverging too much from the original predictive distribution, ρ, since that is likely to affect the coherence and paraphrase quality of the generated text. Following previous work (Jaques et al., 2017; Ziegler et al., 2019), we add a KL divergence penalty, D_{KL}, to discourage this behaviour. The modified reward function, after the baseline and the KL divergence term are incorporated, becomes:

$$R(x, x') = r(x, x') - b(x) - \beta D_{KL} \tag{3.4}$$

where β is a scaling constant, and:

$$D_{KL} = \mathbb{E}_{x \sim \mathcal{D}, x' \sim \pi(x)}[\log \pi(x'|x) - \log \rho(x'|x)] \tag{3.5}$$

This leads to the overall loss function:

$$\mathcal{L} = -R(x, x') \log \pi(x'|x) \tag{3.6}$$

Finally, to prevent longer sequences from being unfairly penalised, we normalise the log probability terms $\log \pi(x'|x)$ and $\log \rho(x'|x)$ in (3.5) and (3.6) by dividing each by the generated sequence length, T.

3.3.3 Paraphrase reward

Let f be the probability output by the victim classifier, x be the original example with label y, and x' be a paraphrase. Let $V(x, x') = f(x)_y - f(x')_y$ be the degradation in confidence in y that x' induces in f. Then the paraphrase reward to use in (3.4) is:

$$r(x, x') = \max(0, \min(\alpha, \eta \delta(x, x') V(x, x'))) \tag{3.7}$$

where α is an upper bound, η a scalar multiplier, and $\delta(x, x')$ a Dirac delta function that is 1 if the constraints (Section 3.3.5) are met, and 0 otherwise.

3.3.4 Reward baseline

As shown in Equation 3.4, the gradient estimator requires a baseline b for the reward. We use a per-example baseline, $b(x)$, and define it as the average reward of the set of adversarial example candidates generated for each x in the training set. The baseline is updated in each validation phase (see Figure 3.2b). Intuitively, the $b(x)$ baseline is high when the model can easily generate adversarial examples for x, and low when it cannot.

3.3.5 Adversarial example constraints

In addition to switching the predicted label, an adversarial example should both preserve meaning (Michel et al., 2019) and be linguistically acceptable. We enforce these principles by using the following constraints:

Label invariance. The original and paraphrase must have the same ground-truth label. Since the ground-truth label of the paraphrase is latent, this constraint is failed if the paraphrase contradicts the original with a probability ≥ 0.2 according to a natural language inference pre-trained model.

Is semantically consistent. The original and paraphrase must have (broadly) the same semantic content. To assess this, we extract sentence embeddings of both using a pre-trained Siamese-BERT model (Reimers and Gurevych, 2019), compute their cosine similarity, and impose a minimum threshold of 0.8.

Is linguistically acceptable. Paraphrases should be acceptable sentences. This constraint is met only if the generated sentence is deemed linguistically acceptable with a probability ≥ 0.5, according to a pre-trained language model.

Through trial and error, we decided to also introduce two additional constraints to prevent the generation of undesirable solutions:

The sentence length is similar. To prevent the generation of very short sentences, we require the original and paraphrase to have sentence length within 30 characters of each other.

Avoids linking contrast phrases. Regardless of the true class, the model can "soften" the generated paraphrase by starting or ending it with a linking contrast phrase, such as "however" or "nonetheless" (see Table 3.5). To encourage the generation of more interesting solutions, we disallow this behaviour, unless the original example itself starts or ends with that phrase.

3.4 Experimental Setup

3.4.1 Datasets

The experiments have been carried out on two English sentiment analysis datasets, each consisting of sentences or short text fragments. The first is the Rotten Tomatoes dataset (Pang and Lee, 2005) which contains extracts of movie reviews with sentiment labelled as either positive or negative. We have used the predefined training, test and validation splits. The second is the Financial PhraseBank dataset (Malo et al., 2014) which contains financial news fragments with sentiment labelled as positive, neutral or negative. We

have used the dataset version with at least 50% annotator label agreement, and randomly selected 10% of the data as the validation set and 10% as the test set.

For both datasets, we have excluded the training examples that the victim model classified incorrectly, as they could be said to be already "adversarial". We have also only included examples with 32 tokens or fewer, since the pre-trained paraphrase model had been trained on sequences in that range. The final datasets are relatively small and, as such, are challenging to train on effectively.

3.4.2 Hyperparameters and design choices

Since design choices significantly impact the attack success rate of the fine-tuned model, in the experiments we have explored the impact of two: the decoding sampling temperature used during training, which controls the exploration of the agent; and the decoding method used for inference and evaluation, which affects the diversity and quality of the generated candidate set. All other hyperparameters have been kept constant. Section 3.9 provides a complete list of the hyperparameters and more training details.

3.4.2.1 Decoding temperature during training

During training, we generate the paraphrases using nucleus sampling, with the probabilities returned by a softmax operator. The temperature parameter used in the softmax, which we denote as τ, visibly affects the generated text: higher temperatures produce more randomness and more diverse training examples, but also a lower percentage of valid English sentences. We have therefore investigated two values of τ: a low-temperature condition of $\tau = 0.85$, and a high-temperature condition of $\tau = 1.15$. We found both these values retain sentence semantics and give somewhat diverse examples, while also being fittingly different from each other.

3.4.2.2 Decoding method during evaluation

During evaluation, i.e. both validation and test, we generate a set of $n = 48$ paraphrases per original example, and then check if any are valid adversarial examples based on the constraints. The decoding method used influences the characteristics of the generated text — and consequently, the attack success rate. This is a key design choice and we have therefore investigated four different methods:

1. *Sampling.* We have used nucleus sampling, with top-p at 0.95 and the temperature at 1.

2. *Beam search.* We have set the number of beams to 48, one per generated example.

3. *Low-diversity beam search.* Diverse beam search (Vijayakumar et al., 2018) is a beam-search variant that increases diversity of generated sequences by dividing the beams into groups and encouraging diversity between them. For this condition we have used six beam groups, set the diversity penalty to 1, and again used 48 beams, one per generated example.

4. *High-diversity beam search.* As above, but with 48 beam groups instead of six.

3.5 Results

3.5.1 Attack success rate

First, we have investigated if the proposed approach improved the attack success rate over the original paraphrase model. As mentioned in Section 3.3.1, an attack is counted as successful if at least one example in the candidate set is validated as a true adversarial example, in that it meets the validity constraints while also inducing a misclassification (NB: typically, many more than one are). The attack success rate is then computed as the percentage of successful attacks. In this section, we report the attack success rate for the original and fine-tuned paraphrase models on the test split of the two datasets, across the two temperature settings and the four decoding methods. For all conditions, we have run three runs from different random seeds and chosen the best performing on the validation set.

Results. The results are reported in Table 3.1, showing that the proposed approach has improved the attack success rate across all training conditions. All improvements have been statistically significant ($p < 0.01$) according to a bootstrap test, as recommended by Dror et al. (2018). We have found no clear best between the two temperature values ($\tau = 0.85$ and $\tau = 1.15$), but beam search as the decoding method has reported the highest success rates on both datasets (61.9% on Rotten Tomatoes and 82.0% on Financial PhraseBank). We have also observed a substantial variance in the results across random seeds (see Figure 3.4a). This problem is well known in reinforcement learning (Henderson et al., 2018) and may be mitigated by more sophisticated baselines, such as RELAX (Grathwohl et al., 2018). We have also performed a run on the Rotten Tomatoes dataset by including examples with up to 48 tokens instead of 32, and found no noticeable difference in attack performance.

	Rotten Tomatoes			*Financial PhraseBank*		
Decoding method	*Original*	$\tau = 0.85$	$\tau = 1.15$	*Original*	$\tau = 0.85$	$\tau = 1.15$
Sampling	30.6	41.8	46.8	20.8	72.3	83.6
Beam search	14.5	**85.5**	**85.5**	11.3	82.4	**88.1**
Low-diversity beam search	20.6	43.2	57.7	13.2	79.3	79.9
High-diversity beam search	24.5	66.0	37.3	21.4	76.1	77.4

Table 3.1: Test-set attack success rate (as a percentage) of the original and fine-tuned models. We use τ to refer to the decoding temperature during fine-tuning. The best results for each dataset are in bold. Fine-tuning has improved the attack success rate for all conditions ($p < 0.01$ according to a bootstrap hypothesis test (Dror et al., 2018)). Amongst the evaluation decoding methods, beam search has had the highest success rates, followed by low-diversity beam search.

3.5.2 Comparison with established adversarial attacks

Next, we have compared the proposed approach to results from a range of competitive adversarial attacks. We have created six different attacks using the TextAttack library (Morris et al., 2020b), varying the transformations and the search method, and we also compared against three established adversarial attack recipes: TextFooler (Jin et al., 2020), BAE-R (Garg and Ramakrishnan, 2020) and IGA (Wang et al., 2021a). These attacks use a variety of language models and embeddings, span a range of query budgets (i.e., the

number of victim model queries per example attacked), and have a corresponding range of attack success rates (Yoo et al., 2020). To ensure a fair comparison, each attack abides by the same constraints as the fine-tuned model, while during the iterations the search is not allowed to modify the same word twice or stopwords (more details are provided in Section 3.10). For performance comparison, we have used the best-performing fine-tuned model from Table 3.1 for each dataset, and we have compared it with these attacks in terms of both attack success rate and number of victim model queries.

Results. The results in Table 3.2 show that the fine-tuned model has achieved a much higher attack success rate than the corresponding adversarial attacks for a comparable average number of queries (e.g., 85.5% with 48 queries vs 67.7% with 52 for Rotten Tomatoes). In fact, its attack success rate has been similar to that of the most query-expensive attack tested, despite requiring a fraction of its queries (48 vs 976 on average for Rotten Tomatoes and 1331 for Financial PhraseBank). More so, the fine-tuned model has been able to generate not one, but many successful adversarial examples per original, averaging 19.6 out of 48 for Rotten Tomatoes and 27.6 for Financial PhraseBank. This staggering difference directly stems from the inherent design advantages of the seq2seq approach.

	Rotten Tomatoes		*Financial PhraseBank*	
Attack method	Success %	Avg Queries	Success %	Avg Queries
Adversarial attacks (see Section 3.10)				
LM-based, low-budget	39.6	42	39.0	58
Embedding-based, low-budget (TextFooler)	67.7	52	50.9	72
LM-based, low-budget (BAE-R)	65.2	53	60.4	76
LM-based, medium-budget (variant 1)	69.4	282	69.2	290
Embedding-based, medium-budget	79.4	327	64.8	498
LM-based, medium-budget (variant 2)	69.1	425	69.2	463
LM-based, high-budget (variant 1)	77.2	790	75.5	697
Embedding-based, high-budget (IGA)	86.9	512	76.1	860
LM-based, high-budget (variant 2)	93.9	976	91.2	1331
Fine-tuned model		Queries (Avg Successes)		Queries (Avg Successes)
Eval: beam search, train: $\tau = 1.15$, best run	85.5	48 (19.6)	88.1	48 (27.6)

Table 3.2: Comparison of the best-performing fine-tuned models against a range of other adversarial attacks. LM: language model; variants 1, 2: see Section 3.10. For these attacks we show the average number of victim model queries needed to find a single adversarial example. In contrast, the proposed fine-tuned model performs a fixed number of victim model's queries (48) and generates multiple adversarial examples. The rows are sorted by increasing Avg Queries value for the Rotten Tomatoes dataset.

3.6 Human Validation

To assess if the proposed approach maintains label invariance, we have performed a small-scale human validation.

The assessment has used samples from the Financial PhraseBank test set and compared adversarial examples from the fine-tuned model, the original paraphrase model, and the most successful compared adversarial attack. Details of the methodolgy of the evaluation are given in Section 3.11.

Results. The fine-tuned model and the compared attack have been able to retain the same ground-truth label as the original example at approximately similar rates (59% and 50%, respectively). The original paraphrase model has been able to retain the ground-truth label in all cases (as expected) but at the price of a drastically lower success rate.

The results show that the fine-tuned model has degraded paraphrasing capability compared to the original paraphrase model, but it still has achieved higher label invariance than the compared approach. These results should be judged alongside the success rate, which has been approximately 8 times higher for the fine-tuned model than the original paraphrase model on the Financial PhraseBank test set (Table 3.1). It is also worth noting that label invariance rates vary considerably in the literature, foremost because there are no "standard settings" to measure them across papers. The rates depend on many factors, such as the instructions given to the evaluators, their harshness or lenience, the use of crowd-sourcing platforms, the datasets used, etc. The most important requirement is that the evaluation is applied equally to all compared approaches, as we do here.

3.7 Discussion

3.7.1 Extending to other text classification tasks

To see if our approach generalises to classification datasets outside sentiment analysis, we have tested the approach on the Text REtrieval Conference (TREC) question-type classification dataset (Li and Roth, 2002). This dataset classifies questions as one of six categories, each labelled as the type of answer that is expected (e.g. *Person*, *Description*). For this experiment, we have used the best set of experimental parameters identified in Section 3.5 and Table 3.1, i.e., beam search decoding and temperature of 1.15. We have also tested the approach against two different victim classifiers, the first using a BERT model (Devlin et al., 2019) and the second a DistilBERT one (Sanh et al., 2019), each fine-tuned on the given dataset before being subjected to the attacks. Each experiment has been run from three different random seeds and the average results are reported. For performance comparison, we have compared our approach against two probing token-modification attacks from Table 3.2: the first (LM-based, low-budget (BAE-R)) is an attack that requires a comparable amount of victim model queries, and the second (LM-based, high-budget (variant 2)) is the highest-performing of the compared approaches from Table 3.2. As a baseline/lower bound, we have also included the original paraphrase model (Untrained model). The results are presented in Table 3.3, showing that our model has been able to achieve the highest success rates and the largest amounts of successful adversarial examples per query. In addition, the trends have not been significantly affected by the choice of victim model. We have also performed a run by including input examples with up to 48 tokens instead of 32, and found no noticeable difference in performance. Overall, this gives evidence to the ability of our approach to generalise well to classification tasks other than sentiment analysis, and across different base models.

In addition, in Table 3.4 we present three qualitative examples of attacks generated by the various approaches. The examples show that the two compared token-modification approaches have somehow distorted the original meaning and compromised linguistic acceptability in some cases, while the proposed model has been able to retain the meaning more closely and generate linguistically-correct attacks in all cases. In addition, in the second example the proposed model has been to generate a much more articulate

	BERT		*DistilBERT*	
Attack method	Success %	Avg Queries	Success %	Avg Queries
Adversarial attacks (see Section 3.10)				
LM-based, low-budget (BAE-R)	88.5	31.8	89.3	30
LM-based, high-budget (variant 2)	91.2	150.1	94.1	129.6
Generative models		Queries (Avg Successes)		Queries (Avg Successes)
Untrained model	42.2	48 (1.9)	47.38	48 (2.3)
Fine-tuned model	95.8	48 (26.6)	97.9	48 (28.3)

Table 3.3: Comparison of the proposed model against two probing approaches and the untrained paraphrase model over the TREC dataset. The victim classifiers use BERT and DistilBERT, respectively (top-level column headings), and the results are the average of three runs from three different random seeds. The results show that our model has achieved the highest success rates and the largest amounts of successful adversarial examples per query.

transformation of the original sentence than those in reach of the two token-modification approaches.

Method	**Text**	**Label**
Original	Where does one find Rider College?	Loc
LM-based, low-budget (BAE-R)	or can one find Rider College?	Enty
LM-based, high-budget (variant 2)	So does one find Rider College?	Desc
Fine-tuned model	How does one find Rider College?	Abbr
Original	How can I get some free technical information on Electric Vehicle?	Desc
LM-based, low-budget (BAE-R)	When can I get some basic technical information on Electric Vehicle?	Num
LM-based, high-budget (variant 2)	When can I get some free technical information on Electric Vehicle?	Num
Fine-tuned model	can someone help me get free technical information about electric vehicles?	Enty
Original	What do the Japanese call Japan?	Enty
LM-based, low-budget (BAE-R)	What do the Japanese mean Japan?	Desc
LM-based, high-budget (variant 2)	What do the Japanese in Japan?	Desc
Fine-tuned model	How do the Japanese call Japan?	Abbr

Table 3.4: Examples of successful attacks generated by the various approaches from samples from the TREC dataset (NB: the untrained paraphrase model did not produce any successful attack from these samples). Note that the compared approaches have somehow distorted the original meaning with their token transformations, while our fine-tuned model has been able to retain it more closely.

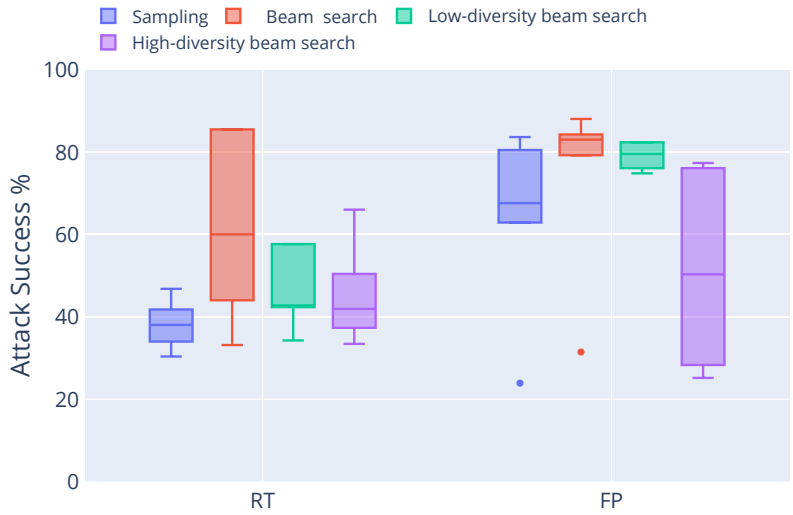
3.7.2 Decoding method analysis

In this section we have compared the different decoding methods used during evaluation/inference in terms of attack success rate, diversity and fluency. Examples of the generated text are in Section 3.12.

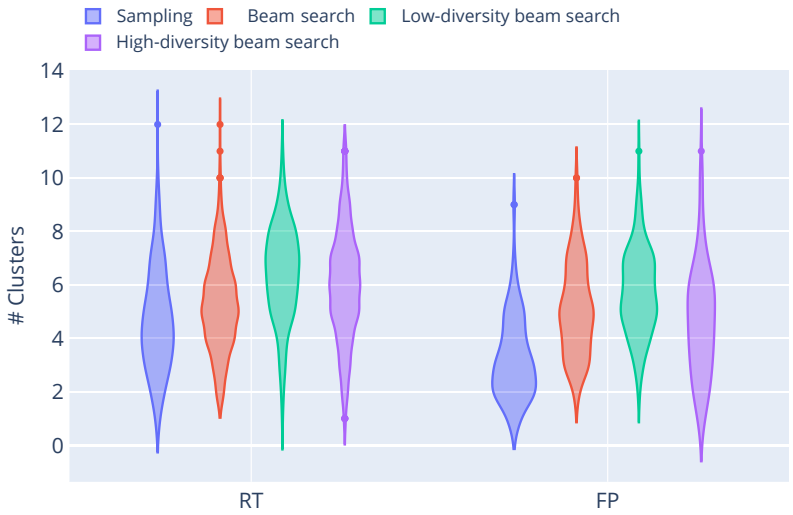
Preprocessing. To measure diversity and fluency, we have selected the two runs with the

highest attack success rate per decoding condition across both training temperatures.[4]

Attack success rate. We found that beam search had the highest attack success rates, followed by low-diversity beam search, although we observed significant variation between runs for all methods (Figure 3.4a).



(a) Attack success rate by decoding evaluation method, across seeds. We see the common RL training problem of high variance across seeds (Henderson et al., 2018). Beam search and low-diversity beam search perform best, on average.



(b) Candidate set diversity of each decoding method, which we measure using a cluster-based score (see Section 3.7.2

Figure 3.4: Attack success rate and diversity of decoding methods. For each graph: RT = Rotten Tomatoes, FP = Financial PhraseBank.

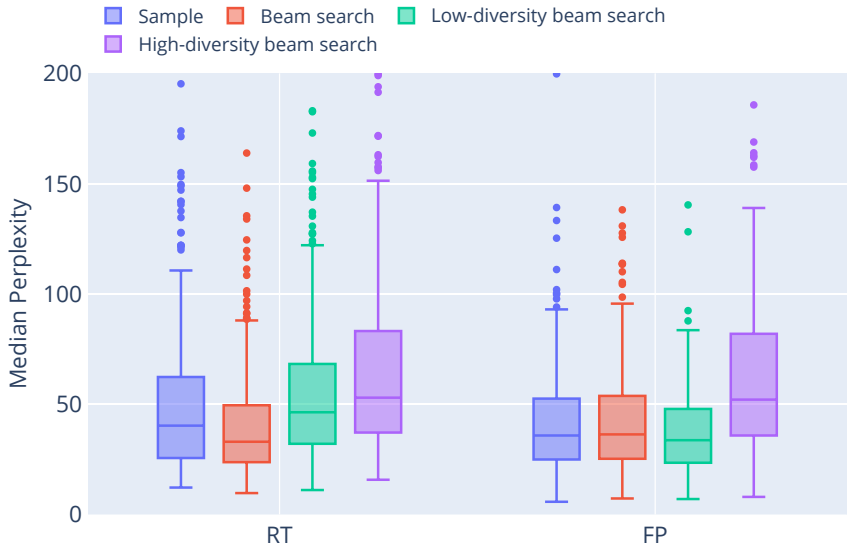
Diversity. To quantify the diversity, we have built a score using the same procedure used for the selection of the adversarial examples for the human validation, described in full in Section 3.11. The score has been simply defined as the number of distinct clusters

[4]We select the best two because there is significant variation in runs between random seeds and it would not be beneficial to compare models that have failed to train properly.

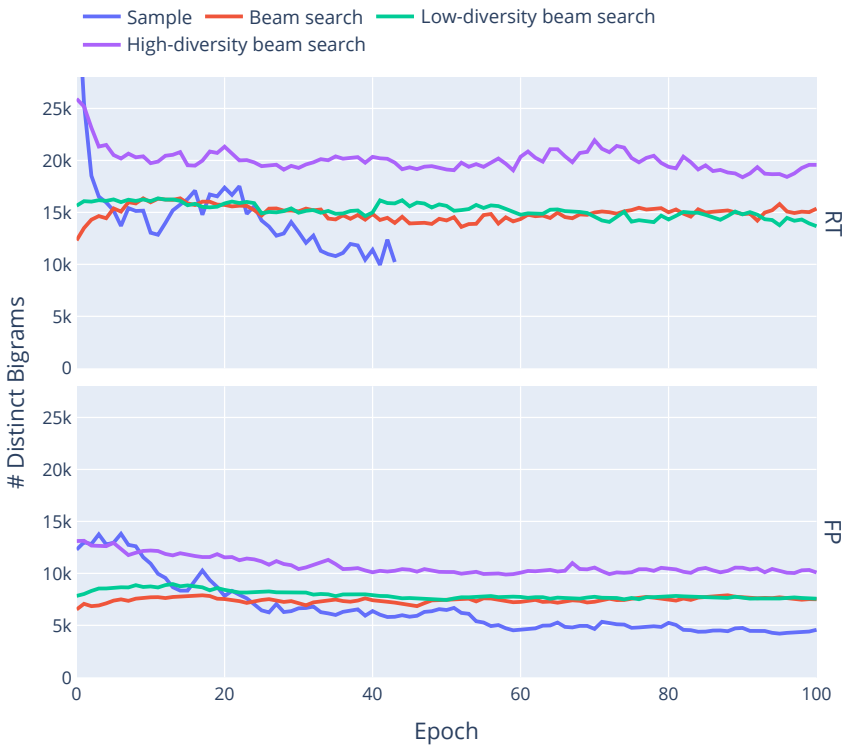
returned by HDBSCAN, with the individual examples not included in any cluster counted as clusters themselves. The results are shown in Figure 3.4.

Overall, low-diversity beam search has generated the most diverse examples, while sampling has generated the least (Figure 3.4b). We have found that, somewhat surprisingly, the high-diversity beam search showed less diversity than the low-diversity beam search, probably because the method tended to generate many "degenerate" examples that were clustered together in our clustering procedure. We have also found that the sampling method generated fewer unique examples on average than the other conditions. We speculate that the token probability over the vocabulary tends to become more concentrated as fine-tuning progresses, as supported by Figure 3.5b.

Fluency. As commonly done (e.g. Wang and Wan (2018)), we have used language model perplexity as a proxy for fluency. We have also used the number of unique generated bigrams performing evaluation on the training set after every epoch as a further measurement of diversity. The results are summarised in Figure 3.5 and show that overall, fluency was similar for all methods except high-diversity beam search.



(a) Median perplexity of the generated candidate sets, with examples combined from the top two runs of each decoding method. Three of the methods have been approximately comparable, while high-diversity beam search has consistently produced the least fluent candidates.



(b) Average number of distinct bigrams generated per epoch performing evaluation on the training set. High-diversity beam search (in purple) has consistently generated more unique bigrams than the other methods. The *sampling* decoding method has displayed a marked decrease in diversity along the epochs, while the others have remained approximately constant. These results confirm the expected trade-off between fluency and diversity.

Figure 3.5: Fluency scores for the various decoding methods. RT = Rotten Tomatoes, FP = Financial PhraseBank.

Overall. Finding the most effective method for this task is subjective, but it can be argued that low-diversity beam search has achieved the best results by balancing attack success rate, diversity, and fluency.

3.7.3 Learned strategies

During training, the model has displayed a wide range of different generating behaviours, such as phrase shuffling, synonym swapping and double negatives, a selection of which is shown in Table 3.5. Some behaviours have proved compatible with our validity constraints, others have violated them. In the initial stages of our research, we had to adjust the reward function repeatedly to disallow some unwanted behaviours, leading to the additional constraints of Section 3.3.5.[5] Some behaviours, such as ignoring grammar, can be considered exploitations of the individual components of the reward function (i.e., "reward hacking"). In addition, the constraints have not been able to perfectly filter all the actual adversarial examples. However, since the components and the constraints are simply proxies for human preferences, the reward can still be effective so long that these behaviours remain limited.

Transformation	Example
Original	safe conduct, however ambitious and well-intentioned, fails to hit the entertainment bull's-eye
Genuine paraphrase	safe conduct might be ambitious and well-intentioned, but it misses the entertainment bull's-eye
Synonym swapping	safe conduct, however ambitious and well-intentioned, fails to strike the entertainment bull's-eye
Case changes	safe Conduct, however Ambitious And well-intentioned, fails to hit the entertainment bull's-eye
Adding/removing punctuation	safe conduct however ambitious and well-intentioned, fails to hit the entertainment bullseye..:;:
Ignoring grammar	safe conduct ambitious well-intentioned bull's-eye
Phrase shuffling	safe conduct, fails to hit the entertainment bull's-eye, however ambitious and well-intentioned
Contradictions	safe conduct, however ambitious and well-intentioned, hit the entertainment bull's-eye
Sentence truncation	safe conduct, however ambitious and well-intentioned
Padding to max length	safe conduct, however ambitious and well-intentioned, fails to hit the entertainment bull's-eye and and and and and and
Very short sentences	safe conduct.
Inserting phone-numbers	safe conduct, however ambitious and well-intentioned, fails to hit the entertainment bull's-eye 888-739-5110 888-739-5110
Using Unicode characters	safe cond©$_{\sqrt{}}$t, however ambitious and well-intentioned, fails to hit the entertainment bull's-eye.
Adding linking contrast phrases	although safe conduct, however ambitious and well-intentioned, fails to hit the entertainment bull's-eye, nonetheless
Repeating phrases	safe conduct, however ambitious and well-intentioned, fails to hit the entertainment bull's-eye, entertainment bull's-eye
Change language of fragment	safe conduct, however ambitieux et well-intentioned, fails to hit the entertainment bull's-eye
Rhetorical question	safe conduct, however ambitious and well-intentioned, fails to hit the entertainment bull's-eye - but why?
Double negatives	safe conduct, however ambitious and well-intentioned, fails to fail to hit the entertainment bull's-eye

Table 3.5: Examples of the model's generating behaviours during training. Rather than only paraphrasing, the model has exhibited a variety of different behaviours. We have introduced two additional constraints to disallow the behaviours of generating very short sentences and adding linking contrast phrases, but we have allowed the others.

3.8 Conclusion

This paper has proposed an approach for generating adversarial attacks for a text classifier based on fine-tuning a seq2seq paraphrase model. The proposed approach fine-tunes the paraphrase model with a novel reward function that encourages misclassifications in the victim model while simultaneously ensuring that the generated attacks abide by a set of validity constraints. The experimental results over three datasets and two victim classifiers have shown that the fine-tuned model has been able to produce many more adversarial examples than the original paraphrase model. It has also proved much more effective and efficient than a set of compared attacks in terms of success rate and number of successful attacks/number of queries trade-off. Future work could include exploring more efficient

[5]This analysis was carried out solely on the training set.

and stable training algorithms, incorporating actual human preferences into the reward objective, and experimenting with different pre-trained seq2seq models, such as those for style transfer or dialogue generation.

3.9 Training Details

This section provides all the details of our training setup.

We have used an AdamW optimiser with early stopping, using the attack success rate of the validation set as the metric, and stopping once the metric dropped below the running median, or when a maximum number of epochs was reached (100 for Rotten Tomatoes, 200 for Financial PhraseBank). We have not done any layer freezing during training as we noticed that it tended to reduce performance.

After the sample selection from the datasets, the splits contained the following number of examples: Rotten Tomatoes — training: 2972, validation: 367, test: 359; Financial PhraseBank — training: 1370, validation: 167, test: 159. The hyperparameters that have been kept constant across all experiments are listed in Table 3.6. The details of the various models — paraphrase model, victim models, and reward component models — are given in Table 3.7.

The GPU model used for training has been an NVIDIA Quadro RTX 6000 with 24GB RAM. Each training run has used a single GPU and the runtime varied from around 4 hours to around 64. The runtime depended heavily on the decoding method (*sampling* was the fastest) and the maximum number of epochs. The training could be potentially sped up by reducing the number of the generated paraphrases during validation. In addition, using a GPU with more memory would allow increasing the batch size, which would also reduce the training time considerably. Occasionally, for some random seeds and conditions, we have run out of memory on the GPU, most likely due to unusually long predictions; repeating the runs with different seeds has always circumvented this issue.

3.10 Compared Attacks

We have created a variety of attack types using the TextAttack library of Morris et al. (2020b). For all attacks, we have set the goal function as untargeted classification (i.e., attacking the correct label with any other class). We have varied the transformation, the search method, and some hyperparameters of the algorithms, and used the sets of constraints of Section 3.3.5. We have also used two extra constraints commonly used to improve the effectiveness of the search (e.g. (Cheng et al., 2020; Jin et al., 2020)): the attacks have not been allowed to modify the same word twice, nor modify stopwords. Tables 3.7 and 3.8 provide full details.

3.11 Human Validation Methodology

In this section we present the methodology used for the human validation.

Annotators. The validation has used three annotators, one of which was a native English

Hyperparameter	Value
General	
LR	1×10^{-4}
Batch size	32
Gradient accumulation steps	2
Max paraphrase length	48
Min paraphrase length	3
Max original length	32
Padding multiple	8
Training generation	
Generated sequences per original	1
Top-p	0.95
Eval generation	
Generated sequences per original	48
Top-p (S)	0.95
Temperature (S)	1
Number of beams (BS, DBS)	48
Diversity penalty (DBS)	1
Reward function	
Reward bounds	[0, α = 10]
Victim degradation multiplier (η)	35
D_{KL} scaling coefficient (β)	0.4
Character difference threshold	$\pm$30
Cosine similarity threshold	0.8 ($\geq$)
NLI contradiction threshold	0.2 ($\leq$)
Linguistic acceptability threshold	0.5 ($\geq$)

Table 3.6: Hyperparameters used for training and evaluation across all experiments. S refers to sampling, BS to beam search, DBS to the two diverse beam search conditions, and NLI to natural language inference. The signs for the thresholds indicate the direction needed to meet the condition.

Purpose	Size	Identifier
Paraphraser	892	prithivida/parrot_paraphraser_on_T5 (Damodaran, 2021)
Victim (RT)	268	textattack/distilbert-base-uncased-rotten-tomatoes
Victim (FP)	329	mrm8488/distilroberta-fine-tuned-financial-news-sentiment-analysis
Victim (TREC) - BERT	433	aychang/bert-base-cased-trec-coarse
Victim (TREC) - DistilBERT	263	aychang/distilbert-base-cased-trec-coarse
Acceptability	47	textattack/albert-base-v2-CoLA
STS	134	sentence-transformers/paraphrase-MiniLM-L12-v2
Contradiction	54	howey/electra-small-mnli

Table 3.7: The models used in this paper. We used small, distilled models to increase training speed and because of our GPU memory requirements, but larger models would give better performance. As above, RT stands for Rotten Tomatoes, FP for Financial PhraseBank. All identifiers refer to models on the Hugging Face Model Hub. Size is given in MB.

Attack identifier	Transformations	Search method	*mc*	Parameters
LM-based, low-budget	Word replacement using language model	Beam search	5	$bw = 2$
Embedding-based, low-budget (TextFooler)	Word replacement using CF embedding	WIR (delete)	50	-
LM-based, low-budget (BAE-R)	Word replacement using language model	WIR (delete)	50	-
LM-based, medium-budget (variant 1)	Word replacement using language model	Beam search	25	$bw = 5$
Embedding-based, medium-budget	Word replacement using CF embedding	Beam search	25	$bw = 5$
LM-based, medium-budget (variant 2)	Word replacement using language model	Genetic algorithm	25	$p = 60$, $mi = 20$, $mr = 5$
LM-based, high-budget (variant 1)	Word replacement using language model	Beam search	50	$bw = 10$
Embedding-based, high-budget (IGA)	Word replacement using CF embedding	Genetic algorithm	50	$p = 60$, $mi = 20$, $mr = 5$
LM-based, high-budget (variant 2)	Word replacement, addition, deletion using language model	Beam search	25	$bw = 5$

Table 3.8: Compared attacks and their parameters. All attacks were generated using the TextAttack package. We have used a DistilRoBERTa language model*. The column mc refers to the maximum number of candidate synonyms allowed. LM refers to the language model, and CF embedding refers to GloVE word embeddings that have been counter-fitted, a procedure introduced in Mrkšić et al. (2016) and commonly used in finding word replacements. WIR (delete) stands for Word Importance Ranking by deletion, a search method where the importance of each token is estimated by deleting it and measuring the true-class confidence drop in the victim model. We use bw to mean beam width, and for the genetic algorithm search parameters, p is the population size, mi the maximum number of iterations, and mr the maximum number of replacements per index.

* https://huggingface.co/distilroberta-base

speaker and two were proficient English second-language speakers.[6]

Models. We have compared the original paraphrase model, the fine-tuned model, and the best-performing baseline algorithm (*LM-based, high-budget (variant 2)* from Table 3.2) based on the attack success rate. For the fine-tuned model we have chosen a run using low-diversity beam search, since in Section 3.7.2 we argue that this has proved the best decoding method overall.

Dataset. We have used 41 original examples from the Financial PhraseBank test set and generated adversarial examples for them. The original examples were split into two groups. The first is where all the three methods were able to produce adversarial examples (18 examples). In the second, the fine-tuned model and the baseline algorithm were able to produce adversarial examples, but the original paraphrase model was not (23 examples). The original examples have been selected so as to have a close-to-balanced class distribution.

Annotator instructions. The aim of the validation has been to assess whether the adversarial examples generated by the three approaches had retained the same ground-truth label as the original, and this question was asked to the annotators (Figure 3.6 shows a screenshot of the instructions). To determine the sentiment, we have used the same instructions as the original annotation of the Financial PhraseBank dataset (Malo et al., 2014). We have used majority voting among the annotators to determine the label invariance.

Filtering algorithm. Both the original paraphrase model and the fine-tuned model can generate up to 48 adversarial examples per original example, which are too many for a

[6] And, for full disclosure, co-authors of this submission. For this reason, the validation has been carried out *blindly*, without knowledge of the generating approach of the examples under assessment.

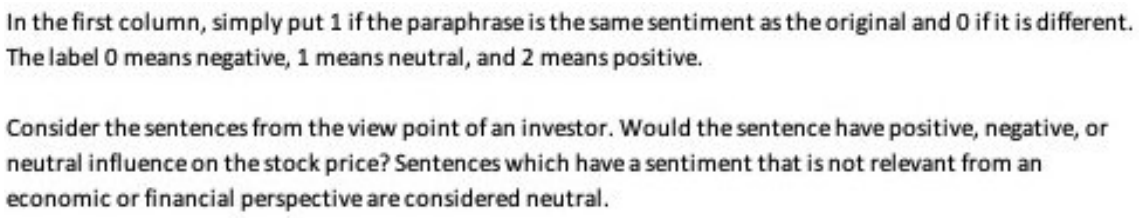


Figure 3.6: Screenshot of the instructions provided to the annotators for the human validation.

human annotator to easily evaluate. To amend this, we have filtered the set of generated adversarial examples to get a smaller representative subset that is less onerous to evaluate. The pseudocode for the filtering is provided in Algorithm 1.

The goal is to end up with a set of roughly 6-12 candidates per original example that approximately preserve the diversity of the larger set. In brief, the algorithm computes sentence embeddings for all candidates, applies a dimensionality reduction algorithm, UMAP (McInnes et al., 2018), and then applies a hierarchical clustering algorithm, HDBSCAN (McInnes et al., 2017). We have eventually sampled a few candidates from each cluster, with the actual number depending on the number of clusters extracted by HDBSCAN.

Algorithm 1 Pseudocode for filtering the generated adversarial examples.

Select all successes $\mathbb{S}$ from candidate set $\mathbb{C} \in \mathbb{S}$
if $|\mathbb{S}| \leq 6$ **then**
    Return $\mathbb{S}' = \mathbb{S}$
else
    Compute sentence embeddings $E(s), \forall s \in \mathbb{S}$
    Reduce dimensionality to get $E'(s), \forall s \in \mathbb{S}$
    Cluster $E'(s)$
    Create $\mathbb{S}'$ by sampling from each cluster
    Return $\mathbb{S}'$
end if

As an example from Table 3.10, the evaluators might consider if the adversarial paraphrase "*In addition he says the company will close a maximum of ten jobs*" has the same sentiment as the original "*In addition , the company will reduce a maximum of ten jobs*". Following the instructions in Figure 3.6, it is likely an investor would consider both sentences as having the same influence on the stock price, and so this would be considered a successful adversarial example, although the semantics are slightly different.

3.12 Examples

Tables 3.9 and 3.10 show examples of the generated text. We present one example per dataset, including all the decoding methods with the best original and fine-tuned models, and the best-performing token-modification attack. For reasons of space, the examples only show eight generated adversarial candidates for each approach.

Each decoding method has exhibited specific trends. Sampling and beam search have generated examples that are very similar to each other, only differing slightly. Addition-

ally, sampling has generated fewer unique candidates than the other methods. Both low and high-diversity beam search have generated good quality candidates, but also many that were ungrammatical or incoherent; and more so for the high-diversity case. This is because sampling introduces randomness, leading to similar outputs by favouring high-probability words, while beam search optimises for sequences that are most probable, often resulting in high overlap among generated candidates. High-diversity beam search, aiming to encourage diversity, tends to push for less probable sequences, which increases the likelihood of incoherence or grammatical errors.

Overall, we notice that the generated adversarial candidates seem to frequently contain consistent phrases or terms, which makes them *universal adversarial examples* (Wallace et al., 2019a). The training procedure clearly learns which phrases affect sentiment across many examples, and the approach inserts them into its generated solutions. This is less desirable than a model that can generate "tailored" paraphrases for each original example. The paraphrasing capability of the fine-tuned model is still remarkable, although at times visibly lower than that of the original paraphrase model. Overall, the fine-tuned model seems to have been able to generate better paraphrases on Financial PhraseBank than Rotten Tomatoes, likely because of its simpler language.

Approach	Generated Text
Original example	suffers from unlikable characters and a self-conscious sense of its own quirky hipness.
Token-modification	suffers from unlikable characters and gains a self-conscious sense of its own quirky hipness .
Sampling, original paraphrase model	suffers from unlikable characters and has a sense of self-conscious hipness. suffers from unlikable characters and a self-conscious sense of its own quirky hipness. suffers from unlikable characters and a self-conscious sense of its own quirky hipness.. suffers from unlikable characters and an in-your-face sense of quirky hipness. suffers from unlikable characters and a self-conscious sense of its own quirky hipness... suffers from unlikable characters and an internal sense of eccentric hipness.. suffers from unlikeable characters and a self-conscious sense of its own quirky hipness. ” suffers from unlikable characters and self-conscious sense of humour.
Sampling, fine-tuned model	Unlikable characters may even have a self-conscious hipness Unlikable characters may even have a self-conscious sense of their own quirky hip Unlikable characters may even have self-conscious hipness Unlikable characters may even have a self-conscious sense of their own hip Unlikable characters might even have an self-conscious hip hop habit Unlikable characters might even have a self-conscious hip hop attitude Unlikable characters might even have a self-conscious sense of their own quirky hip Unlikable characters may even have self-conscious hip-ness
Beam search, original paraphrase model	it suffers from unlikable characters and a self-conscious sense of its own quirky hipness. suffers from unlikable characters and a self-conscious sense of its own quirky hip-hop. suffers from unlikable characters and a self-conscious sense of its own quirky hip-ness. suffers from unlikable characters and self-conscious sense of its own quirky hipness. he suffers from unlikable characters and a self-conscious sense of his own quirky hipness. suffers from unlikable characters and a sense of self-consciousness of its own quirky hipness.. suffers from unlikable characters and a self-conscious sense of quirky hipness.. suffers from unremarkable characters and a self-conscious sense of its own quirky hipness.
Beam search, fine-tuned model	It suffers from unlikable characters and a self-conscious sense of its own quirky hipness but there is something It suffers from unlovable characters and a sense of its own quirky hipness but there is something It suffers from unlikable characters and an unconscious sense of its own quirky hipness but there is something It suffers from unlikable characters and an inner sense of its own quirky hipness but there is something It suffers from unlikable characters that have a sense of their own quirky hipness but there is something It suffers from an unlikable character and a sense of its own quirky hipness but there is something It suffers from unlikable characters and a feeling of its own quirky hipness but there is something It suffers from unlikable characters and a sense of its own quirky hipness but there is some
Low-diversity beam search, original paraphrase model	suffers from unlikable characters and a self-conscious sense of its own quirky hipness. suffers from unlikable characters and a self-conscious sense of its own quirky hipness. ” suffers from unlikable characters and a self-conscious sense of its own quirky hipness. ” suffers from unlikable characters and a self-conscious sense of its own quirky hipness a character that suffers from an unlikable character and a self-conscious sense of its own quirky hipness there is a lack of likeable characters and a fear of its own eccentricity. there is a lack of likeable characters and a fear of its own quirky hips. there is a lack of likeable characters and a sense of its own quirky hipness..
Low-diversity beam search, fine-tuned model	It suffers from unlikable characters but uses a sense of its own quirky hipness It suffers from unlikable characters but includes a sense of its own quirky hip It suffers from unlikable characters but often displays its own quirky hip It comes from unlikable characters but uses a sense of its own quirky hipness It comes from unlikable characters but includes a sense of its own quirky hipness It struggles with unlikable characters but often displays its own quirky hip It also has unlikable characters but is intended for its own quirky hip It struggles with unlikable characters but often displays its own quirky hipness but uses some hip
High-diversity beam search, original paraphrase model	has unlikable characters and a sense of its own quirky hipness. it has unlikable characters and a self-conscious sense of its own quirky hipness. suffers from unlikable characters and an uncanny sense of its own quirky hipness. it has unlikable characters and a sense of its own quirky hipness. the characters are not likable but the sense of their own quirky hipness is a problem. the characters are not likable but the sense of their own quirky hipness is a problem. has unlikable characters and a self-conscious sense of its own quirky hipness.... shows an unlikely character with an unrequited feeling of hipster...
High-diversity beam search, fine-tuned model	The unlikable characters should have a self-conscious sense of hipness Sometimes it should suffer from unlikable characters but sometimes self-conscious senses of hipness Unlikable characters should have a self-conscious sense of hipness This can involve unlikable characters but sometimes self-conscious hipness. It must suffer from unlikeable characters but a self-conscious sense of hipness The unlikeable characters must have a self-conscious sense of hipness The unlikeable character should have a self-conscious sense of its own hipness it will have to suffer from unlikable characters but often quirky hipness

Table 3.9: Examples of generated adversarial candidates for the various approaches for an original example from the Rotten Tomatoes dataset. The original example is in black and has negative sentiment. We use light grey for unsuccessful candidates and green for a successful adversarial example that changes sentiment to positive.

Approach	Generated Text
Original example	In addition , the company will reduce a maximum of ten jobs .
Token-modification	In addition , the company will reduce a maximum requirement of ten jobs .
Sampling, original paraphrase model	In addition, the company can reduce the jobs of up to ten people. In addition the company will cut a maximum of ten jobs. In addition he says the company will close a maximum of ten jobs. In addition, the company will eliminate up to ten jobs. In addition the company will reduce up to ten jobs. In addition the company has cut a maximum of ten jobs. In addition " the company will cut 10 jobs in 10 months. In addition to the job cuts the company will limit itself to ten jobs.
Sampling, fine-tuned model	In addition, the company has succeeded in reducing a maximum of ten jobs. In addition, the company has succeeded in decreasing a maximum of ten jobs. In addition, the company has succeeded in cutting a maximum of ten jobs. In addition, the company has succeeded inreducing a maximum of ten jobs.
Beam search, original paraphrase model	In addition, the company will reduce a maximum of ten jobs. In addition he will reduce a maximum of ten jobs. In addition, the company will reduce a maximum of ten jobs. The company will also reduce a maximum of ten jobs. In addition, the company will cut a maximum of ten jobs. " In addition the company will cut up to ten jobs. In addition, the company will eliminate a maximum of ten jobs. In addition the company will cut a maximum of ten jobs..
Beam search, fine-tuned model	In addition, the company will also be successful in reducing a maximum of 10 jobs In addition, the company will also see success in reducing a maximum of 10 jobs In addition, the company will also see success in reducing a maximum of 10 jobs. In addition, the company will also see success in reducing ten jobs. In addition, the company will also see success at reducing a maximum of 10 jobs. In addition, the company will also have success in cutting a maximum of 10 jobs. In addition, the company will also find success in reducing a maximum of 10 jobs In addition, the company will also find success in reducing a maximum of 10 jobs.
Low-diversity beam search, original paraphrase model	In addition, the company will reduce a maximum of ten jobs. In addition, the company will reduce a maximum of ten jobs.. In addition, the company will reduce a maximum of ten jobs. " In addition the company will cut a maximum of ten jobs. " In addition, the company will reduce a maximum of ten jobs... In addition, the company will reduce a maximum of ten jobs in the region. He said: In addition he would reduce the company's total number of jobs to 10. In addition, the company will reduce a maximum of ten job cuts.
Low-diversity beam search, fine-tuned model	In addition, the company will benefit from reducere ten jobs. This is good for decreasing a maximum of 10 jobs. Along with this reduction of 10 jobs, the company will benefit from reductions. The company will benefit from r$\sqrt{}$©duiing a maximum of 10-jobs. In addition, the company will benefit from reductions for ten jobs. In addition the company will benefit from reductions for the maximum of 10 jobs. As a result the company will benefit from reductions of up to 10 jobs. The company will benefit from reducing ten jobs.
High-diversity beam search, original paraphrase model	The company will also reduce a maximum of ten jobs. A further reduction of ten jobs is planned. " A maximum of ten jobs will be lost in the company. Furthermore, the company will reduce up to ten jobs.. The company will reduce its workforce by up to 10 percent. " It also plans to eliminate at least ten jobs in the company.. Further the company will reduce a maximum of 10 jobs. Additionally the company will lose up to ten jobs.
High-diversity beam search, fine-tuned model	The company also reduced the reduction of a maximum of ten jobs. Furthermore the company will reduce the reduction of ten jobs. This also improved reduction of up to ten jobs. The company also reduced the reduction of ten jobs.. the company will also improve reductions for up to ten jobs. Also the company would reduce a maximum of ten jobs. The company also reduced the reduction of ten jobs... Additionally the firm would also improve the reduction of the maximum of 10 jobs.

Table 3.10: Examples of generated adversarial candidates for the various approaches for an original example from the Financial PhraseBank dataset. The original example is in black and has negative sentiment. We use light grey for unsuccessful candidates, and green and blue for successful adversarial examples that changed the sentiment to positive and neutral, respectively. (NB: the "Sampling, fine-tuned model" approach only produced four unique candidates for this example.)

Chapter 4

XVD: Cross-Vocabulary Differentiable Training for Generative Adversarial Attacks

In this chapter, we aim to address the third research question outlined in Section 1.2.

Abstract. An adversarial attack to a text classifier consists of an input that induces the classifier into an incorrect class prediction, while retaining all the linguistic properties of correctly-classified examples. A popular class of adversarial attacks exploits the gradients of the victim classifier to train a dedicated generative model to produce effective adversarial examples. However, this training signal alone is not sufficient to ensure other desirable properties of the adversarial attacks, such as similarity to non-adversarial examples, linguistic fluency, grammaticality, and so forth. For this reason, in this paper we propose a novel training objective which leverages a set of pretrained language models to promote such properties in the adversarial generation. A core component of our approach is a set of vocabulary-mapping matrices which allow cascading the generative model to any victim or component model of choice, while retaining differentiability end-to-end. The proposed approach has been tested in an ample set of experiments covering six text classification datasets, two victim models, and four baselines. The results show that it has been able to produce effective adversarial attacks, outperforming the compared generative approaches in a majority of cases and proving highly competitive against established token-replacement approaches.

4.1 Introduction

Text adversarial attacks are subtly manipulated inputs to a machine learning model that have the intent of causing erroneous predictions. These manipulations can drastically alter a model's behaviour and represent a significant challenge for the entire field of machine learning. In the context of text classification, adversaries employ a wide range of techniques, from simple token alterations to full training of generative models, each aiming to exploit the model's weaknesses while also preserving the semantic coherence and grammaticality of the text.

The most prevalent adversarial attack strategy is the *token-based* approach, where adversarial examples are crafted through a sequence of token modifications — replacements, additions, or deletions — guided by search methods like beam search, all while maintaining a series of constraints (Morris et al., 2020b). These attacks are simple and effective, but the search method must be run for each example, and the process can prove very time consuming Yoo et al. (2020). Conversely, *generative* approaches train a text-to-text model to directly produce transformations from original to adversarial examples. These attacks, though less studied, can explore a more expansive range of transformations than token-based attacks, and at inference time can rapidly generate a diverse and intriguing array of adversarial examples. The approach is also flexible, with a range of text-to-text models able to be used for this purpose; examples include Generative Adversarial Networks (GANs) (Zhao et al., 2018), paraphrasers (Iyyer et al., 2018), autoencoders (Xu et al., 2021), or style transfer models (Qi et al., 2021). The main drawback of the generative approach is that the model must be trained to generate effective attacks, which can be challenging due to the difficulty of manual supervision and the lack of straightforward training approaches (Wong, 2017).

Adversarial attacks also differ in the amount of assumed access to the classification model (often called the *victim model*). One common assumption is the *black-box* scenario, where attacks only require access to the victim model's outputs, or sometimes the logits (Biggio and Roli, 2018). The opposite is the *white-box* scenario, where the adversary assumes full information access, including gradients, data, loss functions, and model parameters — effectively, the worst-case scenario for an attacked system Biggio and Roli (2018). These assumptions may seem hard to meet in practice, but increasingly they reflect realistic scenarios given the widespread adoption of publicly available machine learning models (such as those found on the Hugging Face Model Hub[1]). On the other hand, developers can use white-box attacks to identify and fix vulnerabilities in their model. In short, studying white-box attacks remains critical.

An intuitive approach for training a white-box generative attack is to link the generative model to the victim model, so as to use the feedback from the victim model as a training signal. However, this signal is not sufficient to ensure all the other properties required of a satisfactory adversarial attack, such as fluency, grammaticality, closeness to non-adversarial examples, and so forth. For this reason, in this paper we propose leveraging a suitable suite of pretrained language models to encourage such properties at training time. To this aim, during the forward pass our generative model receives an original example in input and generates a "soft token" prediction of adversarial example in output, which is then passed to the victim and downstream models for their processing. Softening the prediction ensures that the entire pipeline remains end-to-end differentiable, and able to leverage the training objectives of the downstream modules as an effective adversarial attack loss function.[2] The parameters of the generative model are then updated in the backward pass, while the parameters of the other models are all kept frozen. After training, the generative model is able to generate not only one, but multiple adversarial candidates per original example, simply by using conventional beam search or any other decoding method.

[1] https://huggingface.co/models

[2] The generative model cannot directly pass text to the other models while keeping the training signal differentiable, as it needs either sampling from the token distribution or taking an $argmax$ — both of which are non-differentiable operators.

An immediate challenge to this approach is that the use of soft predictions to permit overall differentiability requires the alignment of the models' vocabularies, which is not trivial to ensure. The simplest workaround is to constrain all the models to share the same vocabulary and tokenisation algorithm. However, this severely limits the choice of pretrained models. Another possible approach is to restrict the vocabularies of all models to their tokens in common (Song et al., 2021). However, this may majorly limit the expressiveness and articulation of the learned adversarial strategies. Overall, the vocabulary alignment between language models still seems to be a partially unresolved issue in the literature.

For this reason, in this paper, we propose an original approach for training a cross-vocabulary, differentiable white-box generative attack that is able to circumvent this restriction. The core components of the proposed approach — nicknamed XVD, from 'cross-vocabulary differentiable' — include: 1) the use of a suitable set of pretrained language models to provide training signals to the adversarial attack generator; 2) the adoption of soft predictions to ensure end-to-end differentiability, and 3) a set of sparse vocabulary-mapping matrices that map tokens between the vocabulary of the generative model and those of the victim and downstream models, allowing complete freedom in the choice of models. The generative model is then trained using a highly configurable, overall loss function that balances text quality with attack strength. In the experiments, the proposed approach has been compared against four baseline methods on six text classification datasets and two victim models. The results show the effectiveness of the proposed approach at consistently generating high-quality adversarial examples across the range of datasets and victim models. In addition, a comprehensive ablation analysis highlights the contributions of the various components and suggests ways for future improvements.

In summary, our paper makes the following contributions:

1. a novel approach for training generative white-box attacks, based on training signals from a set of pretrained language models and a fully differentiable loss function;
2. a vocabulary-mapping module which grants interoperability to any chosen combination of generative, victim or loss component models;
3. extensive experiments over six text classification datasets and two victim models that give evidence to the effectiveness of the proposed approach;
4. a comprehensive ablation and sensitivity analysis that delves into its benefits and limitations.

4.2 Related Work

White-box token-based attacks date back to at least the work of Papernot et al. (2016). Typically, these attacks leverage the gradient signal of the victim model in two main ways. The first is to rank token importance in the original sentence, thus identifying promising attack targets, as demonstrated in Wallace et al. (2020b). The second is to aid in selecting token transformations that best meet adversarial criteria, as shown in various character-level and word-level attacks (Ebrahimi et al., 2018b; Zhang et al., 2019; Liang et al., 2018).

The differentiable model-cascading approach described in Section 4.1 has been explored by several other studies. For instance, Xu et al. (2021) have used an autoencoder as the generative model and examined several modifications to its training process, such

as label smoothing and copy mechanisms, to enhance the effectiveness of the generated examples. Wang et al. (2020b) have proposed incorporating a downstream model which allows the generative model to control the topic of its generated adversarial candidates at inference time. In contrast, Song et al. (2021)'s approach is based on training the generator to output trigger phrases that, when concatenated to an input sentence, induce misclassification in the victim model. In turn, Guo et al. (2021) have proposed learning an example-dependent matrix of token probabilities, which at inference time is sampled to generate adversarial examples. However, none of these approaches has proposed a systematic and configurable solution for training the generative model to satisfy all the desirable properties of an adversarial attack.

In terms of the vocabulary-alignment issue, the works of Xu et al. (2021), Wang et al. (2020b) and Guo et al. (2021) have all acknowledged the problem, but only implemented the shared-vocabulary scenario. Conversely, Song et al. (2021) have constrained the generative model to only output the common tokens of all vocabularies. As observed in the Introduction, neither of these solutions can be regarded as satisfactory.

4.3 Proposed Approach

4.3.1 Overview

We aim to fine-tune a generative model g, with parameters θ and vocabulary V_g, to generate adversarial examples for victim model v, with vocabulary V_v. The approach includes two additional component models for the training objective: a semantic similarity model, s, of vocabulary V_s, and a natural language inference (NLI) model, n, of vocabulary V_n. The parameters of models v, s and n are all fixed, while those of g are the target of the proposed training approach. The complete setup is shown in Figure 4.1[3].

4.3.2 Training

We initialise the generative model, g, with a pretrained paraphrase model as it is already capable of a range of diverse, semantic-preserving transformations. Given an original example x, we employ g to generate an example x' of length T and its corresponding sequence of token probability distributions, which form a matrix P with dimensions $T \times |V_g|$. We then use the token probability distribution matrix, a vocabulary-mapping matrix, and the token embedding matrix of the downstream model to create a weighted average of token embeddings, allowing us to retain the desirable differentiability. Formally, for any component model V_i, with $i \in \{v, s, n\}$, the respective weighted embeddings W_i are computed as:

$$W_i = PM_iE_i$$

where E_i is the token embedding matrix of model i, and M_i is a vocabulary-mapping matrix (described in Section 4.3.3) that maps V_g, the vocabulary of P, to V_i, the vocabulary of model i.

Additionally, to control the diversity of the generated embeddings, we use the Gumbel-softmax reparametrisation trick (Jang et al., 2017), replacing P with a sampled matrix P_b that incorporates Gumbel(τ) noise, where τ is a chosen temperature parameter. Values

[3] All our code will be released in GitHub after the submission period.

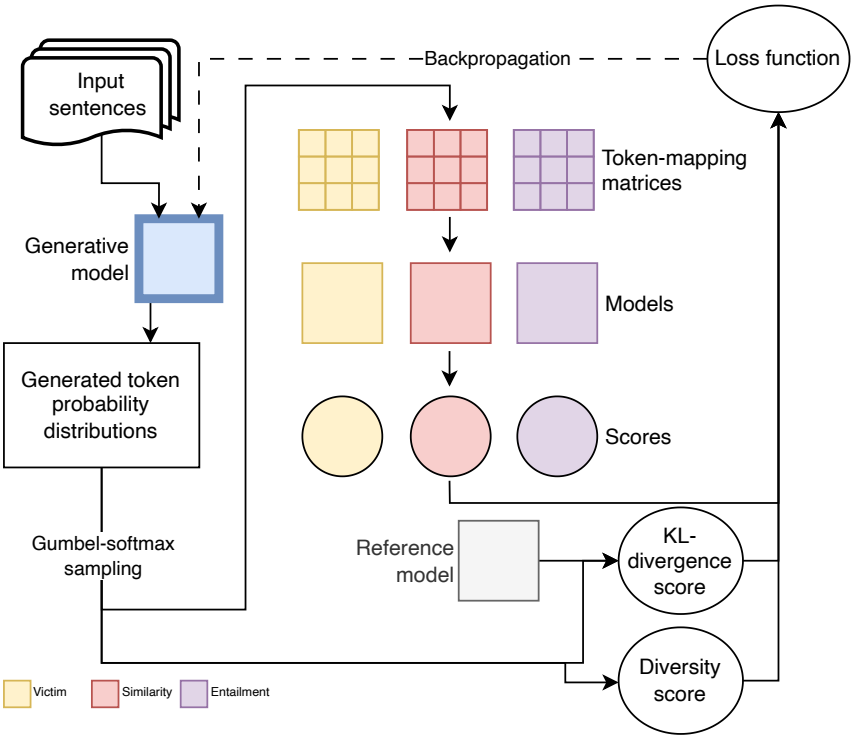


Figure 4.1: The training approach. The loss function is composed of scores from a number of cascaded models (depicted by squares), a KL divergence score using a reference model, and a diversity score. The parameters of the generative model are updated using standard backpropagation.

of $\tau > 1$ make the samples more evenly distributed, while values < 1 concentrate them towards a one-hot distribution. Prior research has also used this technique to increase exploration during training (Xu et al., 2021; Wang et al., 2020b).

After computation, the weighted embeddings $W_i, i \in \{v, s, n\}$ are fed into the component models, and their output scores are used in the loss function (Section 4.3.4). The generative model's parameters are updated via standard backpropagation.

4.3.3 Vocabulary-mapping matrices

In the proposed approach, a vocabulary-mapping matrix, M_i, is constructed to map tokens from the generative model's vocabulary, V_g, to the vocabulary of each component model, noted as V_i hereafter. The matrix has shape $|V_g| \times |V_i|$, and each row is a probability distribution that represents the one-to-many token mapping, with values adding up to 1. This is a large matrix, and to save space we have implemented it as a sparse matrix.

Mapping tokens between vocabularies, each possibly built with a different tokenisation algorithm, is not straightforward. In our implementation, the vocabulary of the generative paraphrase model (of size 32,100) is constructed using the SentencePiece (Kudo and Richardson, 2018) tokenisation algorithm, while the component models' vocabulary (of size 30,522) uses WordPiece (Wu et al., 2016). We have designed our vocabulary mapping for these two tokenisation algorithms since they are among the most ubiquitous in the community. SentencePiece is used in popular models such as T5, XLNet (Yang et al., 2019) and LLaMa (Touvron et al., 2023), while WordPiece is widely used across BERT

variants. The two algorithms have considerable differences, but we have been able to construct a workable mapping with the following rules:

1. Map special tokens (e.g., PAD, EOS, UNK) directly across both vocabularies. Map the extra id tokens in the generator's vocabulary to the UNK WordPiece token (104 matches).
2. Map one-to-one direct matches between SentencePiece start-of-word tokens and WordPiece non-continuation tokens (approximately 9k matches).
3. Map one-to-one direct matches between SentencePiece non start-of-word tokens and WordPiece continuation tokens (approximately 2k matches).
4. Map the remaining SentencePiece tokens one-to-many with WordPiece tokens by passing them as individual strings to the WordPiece tokeniser, stripping any generated special tokens, and assigning equal probabilities to all matches (approximately 22k matches). The few tokens without matches (in practice, special cases like \xad) are mapped to the UNK token.

4.3.4 Loss function

In accordance with the definition provided by Michel et al. (2019), our aim is to create adversarial examples that successfully tweak the predicted labels, yet ensure retention of the text's original meaning alongside linguistic acceptability. To this end, our training objective, $l(x, x')$:

$$\begin{aligned} l(x, x') = \alpha_v t(v(x, x'), \beta_v) + \\ \alpha_s t(s(x, x'), \beta_s) + \\ \alpha_e t(e(x, x'), \beta_e) - \\ \alpha_{KL} t^*(D_{KL}(x, x'), \beta_{KL}) \end{aligned} \tag{4.1}$$

integrates multiple components, each of which encourages the model to generate text with behaviour "fitting" of an adversarial example. These components are:

- $v(x, x')$ represents the 'victim model score', a measure of how much the classifier's confidence in the correct class drops when replacing x to x' in input. This term encourages the generation of misclassified examples.
- $s(x, x')$ is the 'similarity score' between x and x', which is based on the cosine similarity of their sentence embeddings as computed by a pretrained Sentence-BERT model (Reimers and Gurevych, 2019). This term encourages x' to have similar semantic content as x.
- $e(x, x')$ is the 'entailment score', which measures the probability that x's ground-truth label is retained by x', and is approximated with the probability of x entailing x' using a pretrained natural language inference (NLI) model. This term encourages x' to not contradict x.
- D_{KL} is the Kullback-Leibler (KL) divergence between the token probabilities output by the fine-tuned generative model, noted as g, and those of a reference model,

noted as g^*, and taken as the initial pretrained model. D_{KL} is defined as:

$$D_{KL} = \frac{1}{T}\mathbb{E}_{x\sim\mathcal{D},x'\sim g(x;\theta)}[\log p_g(x'|x) - \log p_{g^*}(x'|x)] \tag{4.2}$$

where the generated sequence length, T, is used to normalise the divergence to prevent longer sequences being unfairly penalised. This term encourages the fine-tuned distribution to not deviate excessively from the initial, helping retain the desirable properties of g^*, and helping to prevent text degeneration.

- t and t^* are threshold clipping operators, with $t(a,\beta) = a$ if $a < \beta$, and 0 otherwise, and $t^*(a,\beta) = a$ if $a > \beta$, and 0 otherwise. As such, α and β are hyperparameters that control each term's contribution. These terms encourage the model to consider all components of the objective, and discourage over-optimising for any single component.

The training objective $l(x, x')$ is incorporated into the final *batch-level* loss, L, defined as:

$$L = -\left(\frac{1}{|B|}\sum_{(x,x')\in B} l(x,x')\right) + \alpha_d\, d(B) \tag{4.3}$$

where $d(B)$ is a batch-level diversity score, and α_d its corresponding coefficient. To compute $d(B)$, we first compute the mean of the token embeddings for each generated sentence within batch B. We then calculate the cosine similarity between each pair of mean embeddings using the same model as the similarity score, and compute $d(B)$ as the average, with lower values indicating more diversity. We found that the inclusion of this term can effectively prevent mode collapse, where the model generates repetitive outputs with limited diversity, and encourages variety in the generated examples.

All the terms in the loss function are differentiable by construction, and their convex combination in Equation 4.1 ensures overall differentiability, allowing for efficient minimisation via backpropagation. The coefficients can be adjusted to prioritise different objectives, such as attack strength or fluency.

4.3.5 Validation and early stopping

During fine-tuning, it is important to enforce early stopping to prevent text quality degradation from over-training. To this end, during validation we generate eight candidates per original, using diverse beam search (Vijayakumar et al., 2016). For each candidate, we check if its scores from Equation 4.1 surpass the corresponding β thresholds (or, in the case of the KL divergence, fall below). The validation metric we adopt is the proportion of attacks that have at least one candidate that successfully passes all checks. We calculate the validation metric multiple times per epoch, and halt the training process once it fails to improve over a patience interval, as standard for early stopping.

Dataset	N (trn/val/tst)	Classification task	#cls	Examples
HS	8k/1k/1k	hate speech detection	3	Don't be a [...]" (offensive language)
TREC	4k/1k/0.5k	type of question	6	"When did beethoven die?"(num)
SUBJ	2k/0.5k/0.5k	(subject/object)ivity	2	"[...] harmless diversion and little else" (subj)
RT	3.5k/0.5k/0.5k	sentiment (movies)	2	[...] a not infrequently breathtaking film" (pos)
FP	1.5k/0.2k/0.2k	sentiment (financial)	3	"Operating profit was EUR 11.4 mn [...]" (pos)
Emotion	10k/1k/1k	emotion detection	6	"i be made to feel rotten" (sad)

Table 4.1: Statistics and examples from the datasets used. Column N shows the approximate number of examples in each train/validation/test split, and #cls is the number of classes of the dataset.

4.4 Experimental Setup

4.4.1 Datasets

We have conducted experiments over six diverse English text classification datasets (Table 4.1). The Hate Speech dataset (HS) classifies offensive language in tweets as hate speech, offensive language, or neither (Davidson et al., 2017); the Text REtrieval Conference (TREC) question-type classification dataset (Li and Roth, 2002) and the SUBJ dataset (Pang and Lee, 2004) discriminate between objective and subjective sentences; the Rotten Tomatoes (RT) (Pang and Lee, 2005) and Financial PhraseBank (FP) (Malo et al., 2014) datasets are sentiment analysis datasets of movie reviews and financial news, respectively; and the Emotion dataset classifies text fragments as one of six basic emotions (Saravia et al., 2018). These datasets have been chosen for their diversity and attackable short snippets, with concise statistics and examples presented in Table 4.1.

For each dataset we have used pre-defined train/val/test splits where available (RT and Emotion), and otherwise constructed them by randomly selecting 10% of the data as the validation set and another 10% as the test set (HS, TREC, SUBJ, and FP). For the FP dataset, we have used the dataset version with at least 50% annotator label agreement. For all datasets, we have excluded the training examples that the victim model classified incorrectly, as they could be said to be already "adversarial". We have also only included examples with up to 32 tokens, since the pretrained paraphrase model was trained for sequences in that range.

4.4.2 Models

We have used T5-Base (Raffel et al., 2020), which uses SentencePiece for tokenisation, as our generative model, g. We have evaluated attacks on two victim models: a DistilBERT model (Sanh et al., 2019) and an ELECTRA-trained model (Clark et al., 2020). These are both common BERT variants, each using WordPiece for tokenisation and both small enough in size to fit comfortably on a GPU with a limited memory capacity. Each has been fine-tuned on the given dataset prior to being subjected to attacks.

4.4.3 Baselines

To comparatively evaluate the performance of our model we have used four established baseline attacks, all included in the comprehensive OpenAttack adversarial attack library

Victim Model	Attack	Datasets														
		HS					TREC					SUBJ				
		VSR	Flip	Sim	Flu	Ent	VSR	Flip	Sim	Flu	Ent	VSR	Flip	Sim	Flu	Ent
ELECTRA	BERTAttack	0.37	0.50	0.96	-1.84	0.95	0.33	0.62	0.95	-2.35	0.67	0.46	0.71	0.95	-1.96	0.90
	TextFooler	0.29	0.53	0.92	-2.94	0.80	0.18	0.44	0.93	-2.74	0.49	0.24	0.55	0.94	-2.91	0.69
	GAN	0.00	0.78	0.79	-6.38	0.33	0.00	0.70	0.84	-6.38	0.09	0.00	0.35	0.81	-6.08	0.24
	SCPN	0.12	0.71	0.84	-4.49	0.49	0.25	0.87	0.90	-3.87	0.48	0.28	0.63	0.89	-3.28	0.68
	XVD (ours)	**0.46**	0.80	0.89	-3.30	0.88	**0.58**	0.99	0.92	-3.22	0.80	**0.63**	0.92	0.89	-3.25	0.90
	stdev	*0.02*	*0.04*	*0.00*	*0.28*	*0.02*	*0.02*	*0.04*	*0.00*	*0.28*	*0.02*	*0.03*	*0.02*	*0.00*	*0.12*	*0.01*
DistilBERT	BERTAttack	0.38	0.53	0.96	-1.87	0.94	0.32	0.64	0.94	-2.56	0.69	**0.40**	0.65	0.95	-1.89	0.83
	TextFooler	0.30	0.54	0.93	-2.91	0.79	0.19	0.44	0.93	-2.76	0.52	0.24	0.51	0.94	-2.58	0.64
	GAN	0.00	0.81	0.79	-6.41	0.33	0.00	0.73	0.84	-6.38	0.10	0.00	0.43	0.81	-6.11	0.20
	SCPN	0.13	0.72	0.84	-4.46	0.51	0.27	0.92	0.90	-3.84	0.55	0.23	0.50	0.89	-3.32	0.70
	XVD (ours)	**0.59**	0.82	0.89	-3.14	0.89	**0.37**	1.00	0.89	-3.70	0.65	0.14	0.98	0.82	-4.04	0.79
	stdev	*0.08*	*0.01*	*0.01*	*0.16*	*0.03*	*0.08*	*0.00*	*0.01*	*0.16*	*0.02*	*0.12*	*0.01*	*0.01*	*0.26*	*0.06*

Victim Model	Attack	Datasets														
		RT					FP					Emotion				
		VSR	Flip	Sim	Flu	Ent	VSR	Flip	Sim	Flu	Ent	VSR	Flip	Sim	Flu	Ent
ELECTRA	BERTAttack	**0.47**	0.85	0.96	-1.42	0.79	0.32	0.68	0.96	-1.81	0.56	**0.66**	0.90	0.98	-0.99	0.97
	TextFooler	0.34	0.69	0.96	-2.12	0.69	0.33	0.63	0.94	-2.61	0.71	0.53	0.76	0.97	-1.27	0.95
	GAN	0.00	0.39	0.82	-6.00	0.33	0.00	0.39	0.79	-6.24	0.16	0.00	0.68	0.82	-6.43	0.11
	SCPN	0.28	0.66	0.89	-3.44	0.70	0.14	0.40	0.89	-3.52	0.58	0.37	0.72	0.90	-3.26	0.83
	XVD (ours)	0.30	0.81	0.85	-3.32	0.84	**0.72**	1.00	0.89	-3.12	0.94	0.64	0.97	0.90	-3.24	0.88
	stdev	*0.05*	*0.05*	*0.01*	*0.15*	*0.02*	*0.15*	*0.00*	*0.01*	*0.37*	*0.03*	*0.14*	*0.02*	*0.02*	*0.44*	*0.05*
DistilBERT	BERTAttack	**0.46**	0.89	0.96	-1.40	0.71	**0.43**	0.79	0.96	-1.77	0.82	0.67	0.88	0.98	-0.97	0.98
	TextFooler	0.36	0.70	0.96	-2.10	0.75	0.41	0.76	0.94	-2.60	0.77	0.54	0.76	0.98	-1.28	0.95
	GAN	0.00	0.41	0.82	-5.95	0.32	0.00	0.41	0.80	-6.24	0.18	0.00	0.84	0.82	-6.42	0.11
	SCPN	0.27	0.69	0.89	-3.48	0.63	0.19	0.49	0.90	-3.30	0.63	0.41	0.80	0.90	-3.23	0.80
	XVD (ours)	0.28	0.89	0.85	-3.52	0.77	0.25	0.97	0.86	-3.95	0.72	**0.87**	0.97	0.92	-2.64	0.95
	stdev	*0.10*	*0.02*	*0.02*	*0.23*	*0.03*	*0.15*	*0.04*	*0.01*	*0.21*	*0.21*	*0.00*	*0.00*	*0.00*	*0.02*	*0.01*

Table 4.2: Evaluation of baselines and our approach, XVD, across six datasets and two victim models. For XVD, we report the mean and stdev of each metric across three random seeds (the other approaches are deterministic). We use the following abbreviations: VSR is Validated Success Rate, Flip is the proportion of label flips, Sim is the similarity as measured by BERTScore F1, Flu is fluency as measured by BARTScore, and Ent is the entailment probability measured by an NLI model. Higher is better for all metrics. For dataset abbreviations, see Section 3.4.1.

of Zeng et al. (2021). TextFooler (Jin et al., 2020) and BERTAttack Li et al. (2020c) form the first set of baselines; both are token-replacement attacks that replace individual tokens sequentially in a constrained optimisation process. We have also included two generative attacks that, like our approach, generate adversarial candidates at inference time. The first is a GAN approach (Zhao et al., 2018), and the second is an adversarial paraphraser, named SCPN (Iyyer et al., 2018), that generates syntactically controlled paraphrases.

4.4.4 Candidate selection

At inference time, our fine-tuned model is capable of generating, in principle, an unlimited number of candidates per input example. Nevertheless, for the purpose of fair comparison with the baselines outlined in 4.4.3 that return a single adversarial example per input, we have opted to select only one candidate also from our model.

We begin with the use of diverse beam search (Vijayakumar et al., 2018) to create n candidates for each original example. (A sensitivity analysis of n is presented in Section

4.6.2.) We then compute a 'quality score' for each candidate as $s(x, x') + e(x, x') - D_{KL}(x, x')$, which represents a rough balance of our text-quality objectives. From these scored candidates, we select those that have managed to flip the ground-truth label. Within this subset, we select the candidate with the highest score amongst those that satisfy all validation checks (Section 4.3.5). If none meets these requirements, the highest-scoring candidate is chosen instead.

4.4.5 Evaluation metrics

As an obvious preamble, no ground-truth reference exists for adversarial candidates, and therefore the evaluation has to be orchestrated with adequate and accepted unsupervised metrics. To this aim, we have used five evaluation metrics over the test set of each dataset, and reported the mean values in Table 4.2. The first metric, referred to as *Flip*, is the proportion of instances where the ground-truth label of the original example, predicted correctly by the victim model, has flipped in the prediction for the candidate. This metric has been used almost universally by works in this area (e.g., Li et al. (2020c); Jin et al. (2020); Garg and Ramakrishnan (2020); Li et al. (2021); Formento et al. (2023)). The next three—*Sim, Flu*, and *Ent*— are text quality metrics, and are only computed for examples that have flipped. To assess the semantic similarity between the original and the candidate (*Sim*), we have used BERTScore F1 (Zhang* et al., 2020); to assess the fluency of the candidate (*Flu*), we have used BARTScore (Yuan et al., 2021b), a fluency proxy that uses the text generation probability of a seq2seq model; and for the entailment (*Ent*), we have used the probability that the candidate does not contradict the original in the entailment model. The last metric — the Validated Success Rate (*VSR*) — is the proportion of examples that have successfully flipped the label and also met minimum thresholds across the three text quality metrics.[4] While all automated metrics have inherent limitations, our choice of metrics is both consistent with prior literature and able to provide a thorough assessment of the quality of the adversarial candidate.

Postprocessing. Before metric calculation, each successful attack has been post-processed to begin with a capital letter, end with a period, and have no whitespace around the last punctuation character.

4.5 Results

The results from our experiments are reported in Table 4.2, showing that the proposed approach, XVD, has been able to generate high-quality adversarial examples with notable success rates (VSR). Compared to the generative baseline methods, GAN and SCPN, XVD's performance has proved better for all experimental combinations bar one. XVD has also performed competitively against the best token-replacement baseline, BERTAttack, scoring best for three out of six datasets with the DistilBERT victim model, and for four out of six with the ELECTRA victim model. XVD has also achieved the highest VSR overall (0.87; Emotion dataset). In particular, it has performed the best with both victim

[4]We have used the (fairly relaxed) thresholds of: ≥ 0.85 *Sim*, ≥ -4 *Flu*, ≥ 0.6 *Ent*. These were chosen based on a manual inspection of the text samples. For *Sim* we chose a threshold that would penalise text with large changes in meaning; for *Flu* we chose a threshold based on our own subjective standards; and for *Ent* we observed that all generated texts below this threshold seemed to reliably contradict the original example, or be about a different subject.

models over the TREC dataset, where its flipping rate has proved much higher than that of the other approaches, and over the HS dataset, probably because the token-replacement baselines have struggled to replace its many slang words in the absence of well-defined synonyms. Qualitative examples of the attacks generated by XVD and selected baselines are presented in Table 4.3, showing that the proposed approach has been able to generate more expressive transformations of the original samples, while effectively retaining semantics.

Overall, the proposed approach has proved very strong at label-flipping (Flip) and at retaining the original label (Ent), while intermediate in the fluency (Flu) and similarity (Sim) metrics. This is mainly due to its much broader generative space compared, in particular, to the token-replacement attacks. The proposed approach is also, by design, able to pursue different trade-offs between these properties, thanks to its configurable training objective and generative behaviour. We explore some of these trade-offs in the following section.

As expected, we observed a near-constant runtime for the generative approaches, with XVD's runtime in the order of < 1 s per sample across all datasets.[5] By contrast, we observed a highly variable runtime for the token-modification attacks across datasets, with the runtime increasing if the search space was large and the search instance struggled to find an acceptable attack. For example, BERTAttack's runtime varied from approximately 0.36 s per sample for TREC to approximately 25.2 s per sample for HS.

4.6 Ablation Analysis

We have measured the performance impact of various parameters within our model through a series of ablation studies, using the Financial PhraseBank dataset as reference and testing each configuration using three random seeds. The results are presented in the following subsections.

4.6.1 KL divergence and diversity coefficients

The KL divergence and diversity coefficients (respectively, α_{KL} in Equation 4.1 and α_d in Equation 4.3) define the intensity of their respective regularisers and substantially impact the quality and diversity of the generated text, as shown in Figure 4.2. Increasing the KL coefficient ties the trained model more strongly to the reference model, which in our implementation increases the attack quality at the expense of the label-flipping rate. Conversely, lower values of the diversity coefficient push the model towards samples that are less diverse, while higher values promote diversity per se. Empirically, we have found that that the label-flipping rate has tended to remain constant for a range of diversity values, but the overall text quality metrics have peaked for a value of 10.

4.6.2 Number of generated evaluation sequences

The number of sequences generated during inference, n, directly controls the attack's search space. As n increases, we expect a rise in the label-flipping rate and, after a point,

[5] It is worth noting that our code implementation was far from optimised, with many debugging and logging statements, a small batch size, use of secondary storage etc.

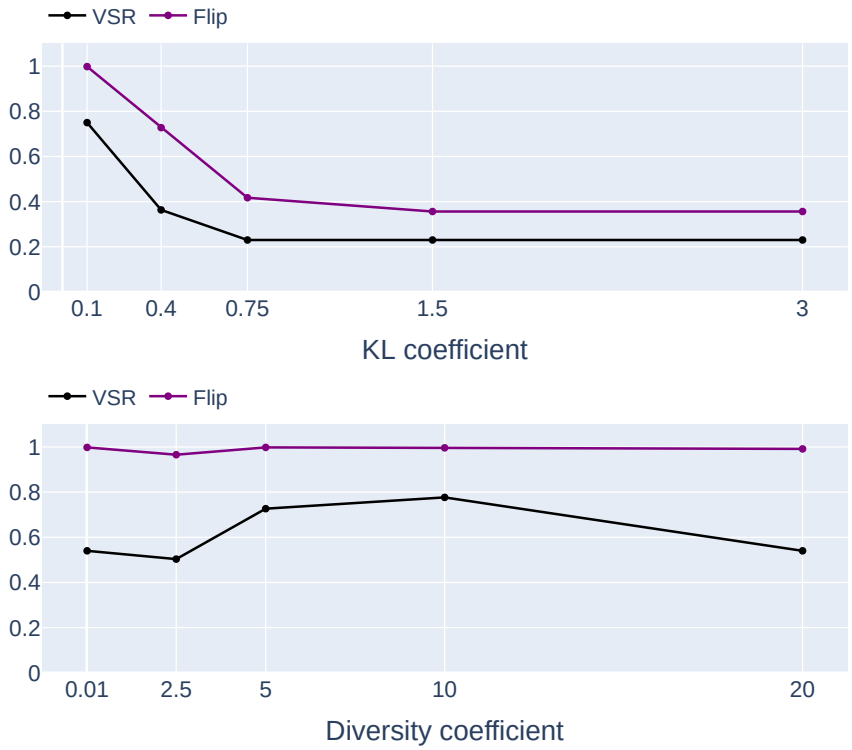


Figure 4.2: Performance as a function of the KL divergence and diversity coefficients.

a decline in text quality metrics. To measure these effects, we have varied n, employing diverse beam search for $n \geq 2$ (with $n/2$ beam groups) and regular beam search for $n = 1$. Our findings, depicted as a plot in Figure 4.3, have confirmed the expected increase in label-flipping rate with larger n. The fluency and similarity metrics have peaked around $n = 4$ before declining, while entailment has remained relatively constant from $n = 8$ onwards. The validated success rate, which compounds the label-flipping rate and the text quality metrics, has improved as n increased, up to a plateau at $n = 32$.

4.6.3 Impact of the vocabulary mapping

To probe the impact of the vocabulary mapping on the performance, we have also carried out an experiment attacking a T5 victim model, which has the same vocabulary as the generative model and dispenses with the need for a vocabulary-mapping matrix. The attacks on the T5 model (on the FP dataset, averaged across three seeds) have resulted in a higher VSR value (0.44) compared to ELECTRA (0.39) and DistilBERT (0.28), implying that the vocabulary-mapping matrix may introduce some performance penalisation. However, this difference could also be due to other reasons, such as the homogeneity between the attacker and the victim model. Since it is not obvious how to precisely excise the impact of the vocabulary mapping from that of the other components, we leave a more exact quantification and possible mitigations to future work.

4.7 Ethical Considerations

The proposed approach potentially raises two main ethical considerations. The first is the potential to generate offensive or inappropriate content. However, this risk, influenced by the training data and the pretraining of the generative model used, is a common chal-

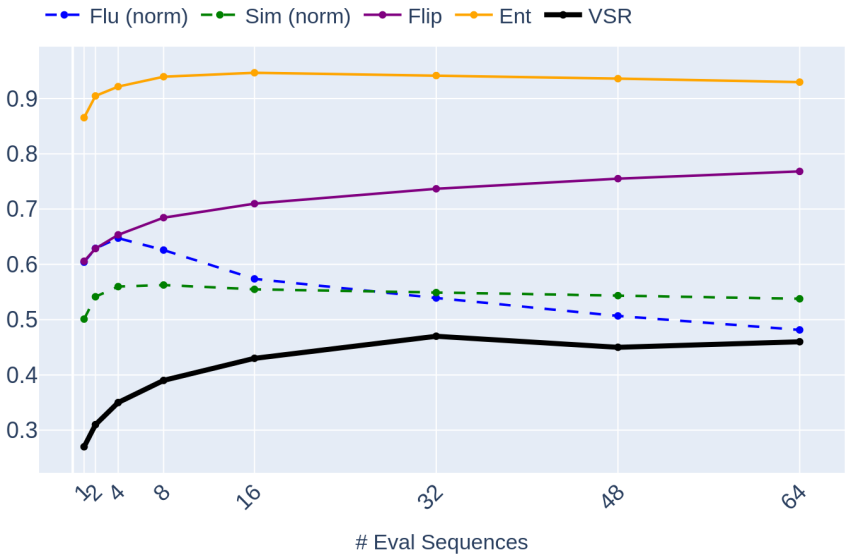


Figure 4.3: Performance as a function of the number of sequences generated per sample (on the FP dataset). The fluency and similarity metrics have been normalised to the [0,1] interval. Higher is better for all metrics.

lenge across text generation models and not specifically our work. The second is that the proposed approach might be used by a malicious actor to deceive or manipulate real-world systems. This risk, however, directly follows from the inherent dual-use nature of adversarial research, where developing methods to defend systems against attacks first requires exploring the attacks themselves. As such, our paper is also helping develop more comprehensive and effective defence strategies.

4.8 Conclusion

This paper has presented an approach for creating flexible white-box adversarial attacks against text classifiers. The key contributions of the proposed approach are its ability to leverage an expressive generative model to generate the attacks, and the integration of dedicated component models in the training objective to encourage their fluency and semantic consistency. In addition, the proposed approach introduces suitable vocabulary-mapping matrices to remap the vocabularies across all the components, allowing the building of a fully-differentiable and highly configurable training objective. Experimental results across six datasets and two victim models have confirmed the viability and effectiveness of the proposed approach, while an ablation analysis has shown the impact of the key parameters on the label-flipping/text quality trade-off. Future research might aim to integrate other components, including possibly human preferences, in the training objective, assess and enhance the performance contribution of the vocabulary-mapping matrices, and adapt the approach to tackle other NLP tasks.

Dataset			Label
TREC	Orig	What is the atomic weight of silver?	Numeric
	BERTAttack	What is the atomic composition of silver?	Description
	SCPN	I'm sorry but that's the atomic weight of silver.	Description
	XVD	Tell me the atomic weight of silver?	Description
SUBJ	Orig	"funny valentine" is about learning what it takes to find true love.	Objective
	BERTAttack	"funny valentine" is about inside what it takes to find true love.	Subjective
	SCPN	I'm learning what it takes to find true love.	Subjective
	XVD	funny valentine's about finding true love.	Subjective
RT	Orig	suffers from unlikable characters and a self-conscious sense of its own quirky hipness.	Negative
	BERTAttack	. from unlikable characters and a self-conscious sense of its own quirky hipness.	Positive
	SCPN	signs of unlikable characters and a self-consciousness sense of its own quirky hipness.	Positive
	XVD	it is characterized by characters who are unlikable and it has a sense of hipness that is self-conscious.	Positive

Table 4.3: Successful adversarial attack examples generated by selected methods. XVD has been able to perform more expressive transformations than the baselines while still retaining sentence semantics.

Chapter 5

A Generative Adversarial Attack for Multilingual Text Classifiers

In this chapter, we aim to address the fourth research question outlined in Section 1.2.

Abstract. Current adversarial attack algorithms, where an adversary changes a text to fool a victim model, have been repeatedly shown to be effective against text classifiers. These attacks, however, generally assume that the victim model is monolingual and cannot be used to target multilingual victim models, a significant limitation given the increased use of these models. For this reason, in this work we propose an approach to fine-tune a multilingual paraphrase model with an adversarial objective so that it becomes able to generate effective adversarial examples against multilingual classifiers. The training objective incorporates a set of pre-trained models to ensure text quality and language consistency of the generated text. In addition, all the models are suitably connected to the generator by vocabulary-mapping matrices, allowing for full end-to-end differentiability of the overall training pipeline. The experimental validation over two multilingual datasets and five languages has shown the effectiveness of the proposed approach compared to existing baselines, particularly in terms of query efficiency. We also provide a detailed analysis of the generated attacks and discuss limitations and opportunities for future research.

5.1 Introduction

Advancements in machine learning have led to models that achieve remarkable performance across a variety of natural language processing tasks. Despite this progress, these models remain susceptible to adversarial attacks, which manipulate inputs to induce incorrect predictions while maintaining linguistic coherence and fluency. In this work we consider text classification, for which a wide variety of adversarial attacks have been developed, ranging from simple character or token replacements to fully generative models.

Text classification models can be developed to process only a specific language (i.e., they are monolingual) or to process multiple languages (i.e., they are multilingual). As processing multiple languages with a single model is more scalable and efficient than training a number of monolingual models, multilingual models are now widely used across many

NLP applications. Despite this, multilingual models have been largely overlooked as victims of adversarial attacks, with the majority of existing attack algorithms being implicitly monolingual and ill-suited to handle multiple languages.

The most common approach for creating text adversarial examples is to leverage combinatorial optimisation in some form. In this approach, an attack takes in input a correctly-classified example and attempts to transform it into an adversarial one by iteratively modifying its individual tokens. The modifications adhere to a set of allowed transformations, such as word insertion or replacement, and are guided by a chosen objective function and search method. Constraints are also imposed to ensure the validity of the transformations, such as requiring semantic similarity to the original input. The process typically stops once a successful adversarial example is found, or the maximum number of attempts is reached. De facto, these attacks solve a separate optimisation problem for each example and are highly effective — provided that they are allowed enough steps, time and computational resources. However, they are typically slow and hard to scale, and the space of the adversarial examples is constrained by the choice of the allowed transformations, which may limit their richness and diversity.

An alternative approach is to use a trainable generative model to produce the adversarial examples. These models learn a mapping from an original example as input to an adversarial example as output based on a training dataset. At inference time, they are able to produce an adversarial attack from the given input in a single forward pass. Since these models do not perform an optimisation process for each input example, they are, in principle, unlikely to match the performance of the transformation-based approach. However, the generative approach mitigates many of its disadvantages: 1) producing the adversarial example is much faster, requiring only a single forward pass per batch of examples; 2) multiple adversarial candidates can be generated simultaneously just by using beam decoding; and 3) it can potentially perform richer and more diverse transformations of the input example.

With this motivation, in this paper we propose an approach for training a generative model to attack a multilingual victim model, under a white-box assumption[1]. We begin with a generative pre-trained multilingual model (specifically, mT5 (Xue et al., 2021)) capable of generating text across more than a hundred languages. Although pre-trained, this model does not yet solve any task, and so we first train it with a multilingual paraphrasing objective. This allows the model to first learn a diverse range of text-to-text transforms in multiple languages. As second stage of training, we fine-tune the generative model with an adversarial training objective. This training objective incorporates both the victim model itself, and also a number of other pre-trained models that help maintain the linguistic properties of the text. Amongst them, a language detection model prevents the generative model from outputting text in languages different from the original's, which is an implicit requirement of this task and a key contribution of the proposed approach. The models are connected with the generative model by matrices that remap their vocabularies, allowing the use of any models in the training objective, while at the same time retaining the end-to-end differentiability of the pipeline. At inference time, the model receives an input in language L and generates adversarial candidates in the same language, provided L is amongst the languages it was fine-tuned in.

To comparatively evaluate the performance of the proposed approach, we have adapted

[1]Which holds for all publicly-distributed models, such as those on Hugging Face and GitHub.

two existing monolingual optimisation attack algorithms to the multilingual case, and used them as baselines. As datasets, we have used two multilingual datasets that incorporate four and five languages, respectively. The experimental results show that our generative model is a fast, diverse and effective attacker, and matches the performance of baselines that run for far longer. In summary, our paper makes the following main contributions:

1. proposes a generative attack for multilingual victim models (a first to the best of our knowledge);
2. proposes a dedicated training objective that leverages the victim model itself and various other modules that promote desirable linguistic properties;
3. makes use of vocabulary-mapping matrices to allow maximum flexibility in the choice of the models and ensure differentiability end-to-end;
4. lastly, our experimental results over two multilingual datasets and five languages show a remarkable comparative performance against the chosen baselines.

5.2 Related Work

Combinatorial optimisation attacks. Combinatorial optimisation attacks are undoubtedly the most common type of attack for text classifiers. Many attacks have been proposed, each defined by their own allowed transformations, search method, and constraints. Well-known attacks include TextFooler (Jin et al., 2020), BERTAttack (Li et al., 2020c), BAE (Garg and Ramakrishnan, 2020) and CLARE (Li et al., 2021). The most recent research has focused on a range of aspects, such as increasing the readability of the adversarial examples (Chang et al., 2023), improving the query efficiency of the attacks (Zhu et al., 2023; Lee et al., 2022) and adapting adversarial attacks designed for images to the text domain (Yuan et al., 2023).

Generative attacks. Although less explored, a number of generative approaches have been proposed. The most relevant to our work are the variations on the white-box, differentiable model-cascading design described in the Introduction. Such work includes Wang et al. (2020b), which has incorporated a downstream model to control the topic of the generated adversarial candidates at inference time, and Xu et al. (2021), which has incorporated several features (e.g., label smoothing and copy mechanisms) to improve the quality of the generated examples. In turn, Guo et al. (2021) have used pre-trained models to learn an example-specific matrix of token probabilities, which can be repeatedly sampled to generate adversarial examples. To avoid the use of vocabulary-mapping matrices (or similar) previous work has largely assumed a scenario where all the component models have to share the same vocabulary, which is a very restrictive assumption. An exception is Song et al. (2021), which has constrained the generative model to only output common tokens to all vocabularies. However, this is also very restrictive.

Multilingual attacks. Most existing text adversarial attacks are monolingual, and predominantly in English. Some work has partially addressed the second restriction by designing attacks for other languages, such as Chinese (Liu et al., 2023; Wang et al., 2022) or Arabic (Alshemali and Kalita, 2019). The only multilingual attack we are aware of is Rosenthal et al. (2021), who have attacked a multilingual question answering system by adding distracting statements to the question. Other multilingual attacks have been

developed for the related task of *code-mixing*, where a single input contains multiple languages (mimicking how multilingual speakers sometimes mix multiple languages when speaking). Code-mixing attacks apply various perturbations to the example: Tan and Joty (2021) have introduced words and phrases through the use of bilingual dictionaries and translation models, and Das et al. (2021) have made phonetically-similar replacements for the Bengali-English and Hindi-English scenarios. However, the code-mixing scenario is different from what we target in our paper: in our case, each example in input to the multilingual classifier is assumed to be in only one language. This allows enforcing a language consistency constraint as a key requirement for the acceptability of the attack.

5.3 Proposed Approach

The training approach fine-tunes a pre-trained generative model g so that it can generate adversarial examples for a victim model v. Besides the victim, the approach incorporates two additional component models into the training objective: a semantic similarity model s and a language detector model l. All models involved have an associated tokeniser with a specific vocabulary, which for model i is denoted by V_i. The parameters of all models are fixed during training, except for g. After fine-tuning, g is able to generate adversarial examples for v. The training setup is shown in Figure 5.1.

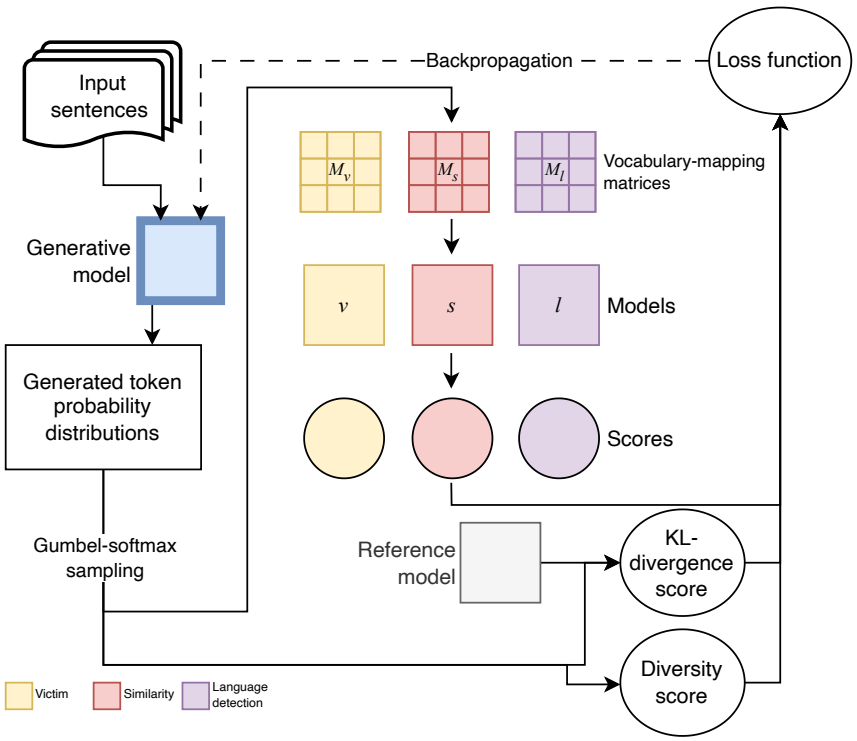


Figure 5.1: The proposed approach. The generative model has been trained using a loss function comprised of a number of factors, including scores from three component models (victim, similarity, and language detection), a KL divergence score, and a diversity score.

5.3.1 Pre-training

For our generative model, we start from a pre-trained mT5-base (Xue et al., 2021) checkpoint. This model has been trained using a span-corruption objective, and has not been

fine-tuned on any downstream task, so the first pre-training step we take is to fine-tune it on a multilingual text paraphrasing training dataset. After this training step, the model is able to generate a range of diverse paraphrases across multiple languages. This gives the starting weights for the generative model g, that is later fine-tuned. We also take a copy of this model and fix its weights to serve as the reference model g^* (see Section 5.3.4).

5.3.2 Fine-tuning

During the forward pass, the input is an original example x which is passed to g, generating output x' with length T. The corresponding sequence of token probability distributions forms a matrix P, with dimensions $T \times |V_g|$.

The generated text cannot be passed to the objective models as is, as that would involve non-differentiable operators (either sampling or argmax). Therefore, we employ a different approach to retain the differentiability of the pipeline: passing a weighted average of token embeddings to the downstream models. This is created by combining P with the token embedding matrix of the downstream model, and a second matrix that maps the vocabulary of the generative model to the vocabulary of the downstream model. We call this second matrix a *vocabulary-mapping matrix* and describe it in detail in Section 5.3.3.

More formally, for any component model V_i, with $i \in \{v, s, l\}$, we have computed the respective weighted embeddings W_i as:

$$W_i = PM_iE_i$$

where E_i is the token embedding matrix of model i, and M_i is the vocabulary-mapping matrix that maps V_g, the vocabulary of P, to V_i, the vocabulary of model i.

As done in prior research (Xu et al., 2021; Wang et al., 2020b), we have used the Gumbel-softmax reparametrisation trick (Jang et al., 2017) and replaced P with a sampled matrix P_b that has incorporated Gumbel(τ) noise. Here, τ is a temperature parameter that controls the entropy of the resulting probability distributions. This step introduces stochasticity into the token distribution with the aim to increase exploration during training.

Afterwards, the weighted embeddings $W_i, i \in \{v, s, l\}$ are passed to their respective downstream models, and the output from each downstream model is used in the loss function (Section 5.3.4). The backward pass computes the chained gradients and updates the parameters of the generative model, using standard backpropagation.

5.3.3 Vocabulary-mapping matrices

Although required for our differentiable training approach, it is not straightforward to map tokens between vocabularies, particularly when they are of different sizes and have been constructed with different tokenisation algorithms. In our implementation, the vocabulary of the generative paraphrase model is constructed using the SentencePiece (Kudo and Richardson, 2018) tokenisation algorithm, while the downstream models' vocabularies use WordPiece (Wu et al., 2016), and differ in content and size. We have been able to construct a workable mapping with the following rules:

1. Map one-to-one all direct matches between SentencePiece start-of-word tokens and WordPiece non-continuation tokens.

2. Map one-to-one all direct matches between SentencePiece non start-of-word tokens and WordPiece continuation tokens.
3. Map any special tokens (e.g., PAD, EOS, UNK) directly across both vocabularies. Map the extra-id tokens in the generator's vocabulary to the UNK WordPiece token.
4. Map the remaining SentencePiece tokens one-to-many with WordPiece tokens using the WordPiece tokeniser, stripping any generated special tokens, and assigning equal probabilities to all matches. Map any remaining tokens (e.g. special cases like \xad) to the UNK token.

The end result is a matrix M_i that maps tokens from the vocabulary of the generative model, V_g, to V_i, the vocabulary of the downstream model. In M_i, each row is a probability distribution that represents the one-to-many token mapping, with the sum of each row being 1, and the vast majority of entries being 0. The shape of M_i is $|V_g| \times |V_i|$, and as this is a large matrix[2], to save space we store it as a sparse matrix.

5.3.4 Loss function

The training objective, denoted as $o(x, x')$, has been designed to balance the adversarial and linguistic factors required by a successful multilingual adversarial attack. It is given by

$$\begin{aligned} o(x, x') = \alpha_v t(v(x, x'), \beta_v) + \\ \alpha_s t(s(x, x'), \beta_s) + \\ \alpha_l t^*(l(x, x'), \beta_l) - \\ \alpha_{KL} t^*(D_{KL}(x, x'), \beta_{KL}) \end{aligned} \tag{5.1}$$

where:

- $v(x, x')$ is the 'victim model score': the degradation in victim model confidence of the true class when replacing x with x'.
- $s(x, x')$ is the 'similarity score' between x and x', given by the cosine similarity of their sentence embeddings from a pre-trained multilingual Sentence-BERT model (Reimers and Gurevych, 2019).
- $l(x, x')$ is the 'language consistency score', given by the degradation in original-language confidence between x and x', using a pre-trained language detection model.
- D_{KL} is a Kullback-Leibler (KL) divergence term, calculated between the token probabilities produced by the fine-tuned generative model g and a reference model g^*. This term prevents the fine-tuned distribution from deviating too much from the reference, which we take as the multilingual paraphrase model before fine-tuning, allowing the beneficial paraphrasing qualities of g^* to be maintained.[3] D_{KL} is defined as:

$$\begin{aligned} D_{KL} = \frac{1}{T} \mathbb{E}_{x \sim \mathcal{D}, x' \sim g(x;\theta)} [\log p_g(x'|x) - \\ \log p_{g^*}(x'|x)] \end{aligned} \tag{5.2}$$

[2] $|V_g|$ alone is 250K.

[3] Due to memory constraints, g^* is only a part of the initial paraphrase model.

In (5.2), the divergence has been normalised by the generated sequence length, T, to prevent longer sequences from being unfairly penalised.

- t and t^* are threshold clipping operators, with $t(a, \beta) = a$ if $a < \beta$, and 0 otherwise, and $t^*(a, \beta) = a$ if $a > \beta$, and 0 otherwise. These operators are intended to encourage a balanced optimisation of the multiple objective components, with hyperparameters α and β controlling the contribution of each component.

As this work extends the method presented in Chapter 4, it is worth noting that this training objective is also similar to the training objective of 4.3.4, but with the 'entailment score' replaced with the 'language consistency score'. The introduced 'language consistency score' (not needed in the previous English-only work) is pivotal to the success of this approach, as later seen in Section 5.4.4. We have opted to remove the 'entailment score' here for simplicity, but the term could easily have been included as well.

This training objective $o(x, x')$ is then incorporated into a *batch-level* loss, L, which is defined as:

$$L = -\left(\frac{1}{|B|}\sum_{(x,x')\in B} o(x, x')\right) + \alpha_d\, d(B) \tag{5.3}$$

where $d(B)$ is a batch-level diversity score, and α_d its corresponding coefficient.

To compute $d(B)$, within batch B and using the same model as for the similarity score, we calculate two cosine similarity matrices: one of the mean of the token embeddings for each generated example, and one of the embeddings of the original input text. We then take the mean of the squared difference of the upper triangular matrices as the diversity penalty $d(B)$. This term encourages the generated examples to maintain a similar batch diversity as for the original examples, as to prevent degeneration during training and improve the variety of the generated examples.

As each individual component in the loss function is differentiable and their combination is linear, the overall loss function is also differentiable and the system can be efficiently optimised using backpropagation. The overall design is flexible, and each coefficient can be adjusted to prioritise different objectives, such as attack strength, textual fluency or language consistency.

5.3.5 Validation and early stopping

The quality of the generated text and its attack strength are somehow inversely proportional to the duration of training: shorter training yields high-quality text with lower attack strength, while prolonged training increases attack strength, yet degrades text quality. Deciding when to stop training is hence an important decision, and we have used the following stopping criteria: during the validation phase, we generate 16 candidates per original example, decoding using *diverse beam search* (Vijayakumar et al., 2016). Each candidate is then given a 'score' based on its adversarial and linguistic qualities:

$$\gamma_v v(x, x') + Q(x, x'),$$

with

$$Q(x, x') = \gamma_s s(x, x') - \gamma_l l(x, x') - \gamma_{KL} D_{KL}(x, x')$$

being a 'text quality' score. The γ coefficients set the balance between attack strength and text quality.

The validation metric we use is the average score across all candidates for the examples in the validation set. We calculate the metric after every multiple of training steps, and halt the training process once the metric fails to improve for a number of steps, as standard in early stopping.

5.4 Experimental Setup

5.4.1 Datasets and models

We have evaluated the approach on two multilingual datasets. The first is the Multilingual Amazon Reviews Corpus (MARC) (Keung et al., 2020), comprised of Amazon customer ratings of the goods they purchased. For our experiments, we have used the English, German, French, and Spanish language splits. The second dataset is the Tweet Sentiment Multilingual (TSM) dataset (Barbieri et al., 2022). This is a multilingual sentiment analysis dataset that categorises a variety of tweets as negative, neutral or positive. For this dataset, we have used the above four language splits, plus Arabic. Finally, to initially train the multilingual paraphrasing objective, we have used the TaPaCo dataset (Scherrer, 2020), selecting the splits that correspond to all languages covered above. For all datasets, we have filtered out all examples longer than 32 tokens, as these longer examples are difficult to paraphrase effectively.

As multilingual victim models, we have fine-tuned a pre-trained multilingual DistilBERT model (Sanh et al., 2019) on the MARC and TSM datasets separately, using only the languages mentioned above. The performance of the resulting trained models is close to the state-of-the-art performance for both datasets, making the adversarial attacks relevant. Full details of datasets and models, including the specific choice of component models, are in Appendix 5.7.

5.4.2 Baselines

As no general multilingual adversarial attack currently exists, for the baselines we have extended CLARE (Li et al., 2021) and BAE-R (Garg and Ramakrishnan, 2020), two existing monolingual combinatorial optimisation attacks, to the multilingual case. We term these new multilingual variants mCLARE and mBAE. Both attacks originally used a BERT masked language model (Devlin et al., 2019) to generate the transform token candidates, and the Universal Sentence Encoder (Cer et al., 2018) to assess sentence similarity. In their multilingual extension, we have replaced them with a multilingual BERT masked language model and the multilingual Universal Sentence Encoder (Yang et al., 2020), respectively. We have also removed language-dependent constraints from the attacks that originally forbade changing stop-words and restricted attack targets to words with a certain part-of-speech tag. The final mCLARE attack is able to replace, insert, or merge individual tokens, while mBAE can only replace.

Since the attack performance for these baselines directly depends on the number of times that the attack is allowed to query the victim model, to get an accurate view of performance we have run the baselines a number of times, each with a differing amount of maximum queries. The number of queries can also be seen as a proxy for the runtime of the method, for which there is a clear correlation (Yoo et al., 2020). However, the actual runtime directly depends on the time per query, which is approximately 10x larger for mCLARE than mBAE.

5.4.3 Candidate selection

At inference time, our generative fine-tuned model is capable of generating, in principle, multiple adversarial candidates per input example by using beam search. However, for fairness we have opted to select only one candidate from our model, because it may be otherwise unfair to compare with the combinatorial optimisation attacks which return only one adversarial example per input. To select our candidate, we start by decoding using diverse beam search (Vijayakumar et al., 2018) to create n outputs for each original example. Then, out of those that flip the label, we select that with the highest text quality score, $Q(x, x')$, as the candidate (see Section 5.3.5 for details).

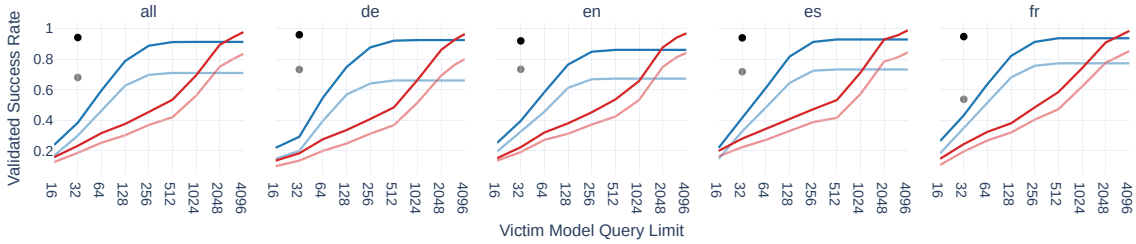
5.4.4 Evaluation metrics

As there no ground-truth references exist for the adversarial example task, prior work has used a number of reference-free metrics to assess each method (e.g. Xu et al., 2021; Garg and Ramakrishnan, 2020), and we also follow the same approach.

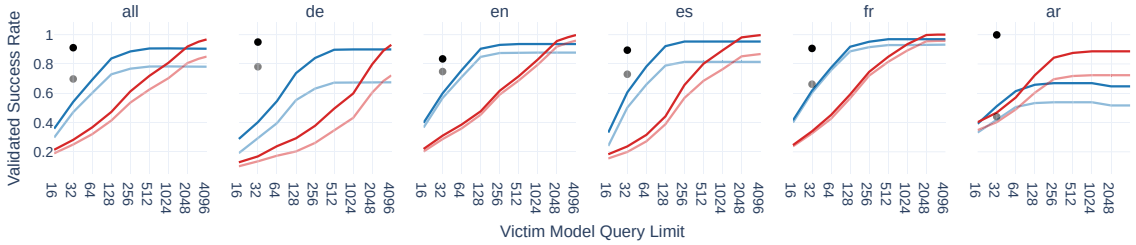
The first metric we use is the label-flip rate: the proportion of attacked examples where the victim model changes its predicted label compared to the original example. The next three are measures of text quality and are only computed for candidates that flip the label. The first of these is an assessment of the semantic similarity between the original and adversarial examples, and the metric chosen is BERTScore F1 (Zhang* et al., 2020), using a multilingual BERT variant as the base model. The second is a proxy assessment of fluency, for which we use BARTScore (Yuan et al., 2021b) which is a measurement of the likelihood of a given output in a chosen multilingual BART model. Finally, the third metric assesses if the generated text is in the same language as the original. We assess this with the confidence output of a language detection model[4]

Eventually, we have combined these metrics into an overall metric, with the aim to identify candidates that do not obviously violate any of the adversarial example principles. We call this metric the Validated Success Rate (*VSR*) and define it as the proportion of examples that have successfully flipped the predicted label while also meeting minimum thresholds on all of the text quality metrics. To set the similarity threshold, we have used as a guide the similarity scores achieved from the multilingual paraphraser before training, which trend around 0.65. We have hence set a threshold of 0.6 to allow the generative model some 'wiggle room' in the text generation. For the language detection, we have empirically set the threshold to 0.5. For the fluency threshold, we have explored how two different choices ("tight", -9, and "loose", -11) affect the overall VSR achieved by each method, as shown in Figure 5.2.

[4]We use a model from the Lingua library (https://github.com/pemistahl/lingua-py)



(a) Results for the MARC dataset.



(b) Results for the TSM dataset.

Figure 5.2: Results for the MARC dataset (above) and the TSM dataset (below), overall and split by language. The x-axis is the maximum number of victim model queries, and the y-axis is the validated success rate (VSR) of the attacks. For all methods, we report VSR values for two different fluency threshold (see Section 5.4.4). The black dots plot the values for the proposed approach (averaged across three random seeds), while the lines plot the values for the baseline methods, with blue for mBAE and red for mCLARE. The results show that the proposed approach has achieved remarkable VSR values in many cases, and has achieved an impressive trade-off between performance and number of queries. At least one of the baselines eventually surpasses its performance in many cases, but only after many more queries (512, 1024 or more).

5.5 Results

The results for each language and for the two different fluency thresholds are shown in Figure 5.2. For the proposed approach, we only plot one VSR value per threshold as a dot in correspondence of query limit = 32, since we use a beam size of 32 and all the beam outputs are assessed against the victim model. For the baselines, we plot full curves with maximum number of queries in the range [16, 4096] (blue: mBAE; red: mCLARE). The larger values correspond to the looser fluency threshold.

The results show that the performance from the proposed generative model has been remarkable. It has outperformed the baselines in the low-query range by large margins, and has remained reasonably competitive for larger numbers of queries. Of the two baselines, mBAE has reported a higher performance for lower numbers of queries, and mCLARE for higher numbers. The performance across languages has been fairly consistent throughout, with the exception of Arabic, for which all methods have struggled. The baselines did more poorly over the German datasets, but this discrepancy can be partially explained by the impact of the document length: the test sets for German are made of shorter docu-

Dataset			Label
MARC	Orig	I love that it keeps the wine cold.	Positive
	mCLARE	I not love that it keeps the wine cold. *Flu*: -7.11 *Lang*: 0.84 *Sim*: 0.96	Negative
	mBAE	I doubt that it keeps the wine cold. *Flu*: -7.60 *Lang*: 0.85 *Sim*: 0.92	Negative
	Ours	In my opinion, it keeps the wine cold! *Flu*: -7.63 *Lang*: 0.78 *Sim*: 0.82	Negative
MARC	Orig	Une batterie neuve qui fait à peine mieux que mon ancienne batterie. Décevant.	Negative
	mCLARE	Une batterie neuve qui fait à peine mieux que mon ancienne batterie. Ravel. *Flu*: -5.24 *Lang*: 0.97 *Sim*: 0.97	Positive
	mBAE	Une batterie neuve qui fait vraiment bon mieux que mon ancienne batterie. Décevant. *Flu*: -4.89 *Lang*: 0.96 *Sim*: 0.99	Positive
	Ours	C'est une batterie neuve qui est bien mieux que mon ancienne batterie. *Flu*: -6.74 *Lang*: 0.99 *Sim*: 0.75	Positive
TSM	Orig	Heute Abend dann noch Haare färben und Sport machen :D	Positive
	mCLARE	Heute Abend dann noch Haare färben und Sport machen :Roman *Flu*: -8.19 *Lang*: 0.88 *Sim*: 0.99	Neutral
	mBAE	Heute Abend dann noch Haare färben und Sport machen :Roman *Flu*: -8.19 *Lang*: 0.88 *Sim*: 0.99	Neutral
	Ours	Was hältst du davon, heute Abend noch Haare färben und Sport *Flu*: -6.58 *Lang*: 0.97 *Sim*: 0.71	Neutral

Table 5.1: Examples of successful adversarial attacks generated by the various methods for three languages. The token changes of the baselines are highlighted in blue, and the values of the fluency, language consistency, and semantic similarity are reported below. The examples show that our method has been able to create richer, coherent adversarial examples than the baselines, which are limited to modifying individual tokens. To generate successful examples, the baselines often employ "tricks" such as inserting negatives (first example), forming ungrammatical sentences (e.g. *bon mieux* for mBAE in the second example, or the misformed emoji *:Roman* in the third) and changing the base language (*Ravel* in the second example).

ments, and the token replacements made by the baselines affect the fluency metric more significantly.

A few, selected qualitative examples of the generated attacks are shown in Table 5.1. These examples show that the generative approach has been able to produce effective adversarial examples by applying richer, sentence-level transforms. By contrast, the baselines' transforms have been more limited, and often not of high linguistic quality.

5.5.1 Sensitivity analysis

The semantic similarity and language detection coefficients in Equation 5.1 (respectively, α_s and α_l) control the intensity of their respective regularisers, and in turn impact the quality of the generated text. We have qualitatively explored the impact of removing these components from the proposed approach, with examples displayed in Tables 5.2 and 5.3. These examples confirm that removing the similarity component reduces the faithfulness to the original example, and that removing the language detection component allows the

Mode	Text	Label
Orig	These raw cashews are delicious and fresh.	Pos
+Sim	I'll like this raw cashew.	Neg
−Sim	I'm going to eat these raw cashiers.	Neg
Orig	I love that it keeps the wine cold.	Pos
+Sim	In my opinion, it keeps the wine cold!	Neg
−Sim	The wine is delicious. I'm going to like	Neg

Table 5.2: Illustrative English examples from training with and without the semantic similarity component. In the first example, *cashew* is changed to *cashier* — a simple change of one subword token that loses the sentence meaning. In the second example, the information about the temperature of the wine is lost.

Mode	Text	Label
Orig	Super, hilft schnell	Pos
+LD	Irgendetwas, das ist schnell.	Neg
−LD	Solch ein ayuda!	Neg
Orig	Ces sachets sentent le plastique.	Neg
+LD	Ces sachets sont pour le plastique!	Pos
−LD	Ces sachetsgefühlt le plastique.	Pos

Table 5.3: Training without the language detection component often leads to foreign words in the generated text. In the first example, the model has directly translated the German *hilft* (to help) to the Spanish word *ayuda*. Similarly, in the second example the French *sentent* (to smell) is translated into the German *gefühlt*.

generative model to "stray" to generate examples containing text in other languages.

5.5.2 Language impact

All methods have struggled with Arabic, and this suggests that specialist adversarial algorithms may be required, such as Alshemali and Kalita (2019). As a plausible reason, Arabic is highly inflected and differs significantly in its morphological and syntactic structure from the other four languages, making it unlikely that the adversarial attacks could effectively exploit their cross-lingual knowledge. Moreover, in the case of the proposed approach the base model, mT5, was originally trained on the mC4 corpus (Xue et al., 2021), which contains far fewer Arabic tokens than for the other languages, and the same applies to the TaPaCo corpus used for the paraphraser training. In turn, mCLARE and mBAE use multilingual BERT as the base language model, which was also trained on less Arabic data than for the other languages and is known to perform poorly compared to monolingual Arabic BERT variants, such as AraBERT (Antoun et al., 2020). Therefore, adversarial performance in Arabic may be improved by adopting dedicated language models. In addition, the BERT tokeniser also introduces a significant amount of redundancy in tokens when tokenising Arabic text (Antoun et al., 2020), and using a specialist Arabic segmenter such as Farasa (Darwish and Mubarak, 2016) could improve performance.

5.6 Conclusion

In this paper, we have presented a white-box generative approach for attacking multilingual text classifiers. The training objective of the proposed approach is very flexible, and incorporates specialised component models to encourage fluency, semantic consistency, and language adherence. To connect the generative model to the component models we have leveraged vocabulary-mapping matrices that allow using models of any vocabulary and retaining the full differentiability of the training objective. Experimental results across two multilingual datasets have confirmed the effectiveness of the proposed approach, particularly in a low number of queries scenario or when richer, sentence-level transforms are desired. Future research in this area might aim to adapt this framework to better target under-resourced or linguistically diverse languages, or to attack models for other multilingual NLP tasks, such as machine translation or cross-lingual classification.

5.7 Training Details

To train the generative model to paraphrase, we have used the TaPaCo dataset, using the splits for the chosen five languages. After filtering out examples longer than our maximum token limit of 32 tokens, we had the following rough number of examples per language: ar: 6.5k, de: 125k, en: 158k, es: 85k, fr: 117k. We then preprocessed these examples to create a corpus of ordered paraphrase pairs (i.e. so that (p1,p2) and (p2,p1) would be two separate paraphrase pairs). We then oversampled or undersampled the pairs to obtain 100k paraphrase pairs per language, and collated them to form our final training dataset. For the MARC and TSM datasets, we have removed all training examples that the victim model classified incorrectly prior to the training procedure, as they could be said to be adversarial already. We have also removed examples longer than 32 tokens, since the paraphrase model was not trained for sequences in this range. For the MARC dataset, we have categorised 1 and 2 star reviews as negative, 4 and 5 star reviews as positive, and discarded the 3 star reviews. Because this is a large dataset we selected a language-balanced subset to use as the training, validation and test sets, choosing 5k samples for the training set and 1k samples each for the validation and test sets. For the TSM dataset, the approximate sizes are in Table 5.4.

Lang	Trn / Val / Test
ar	0.6k / 0.1k / 0.2k
de	1.2k / 0.2k / 0.4k
en	0.6k / 0.1k / 0.3k
es	1.0k / 0.1k / 0.3k
fr	0.9k / 0.1k / 0.4k
all	4.3k / 0.6k / 1.6k

Table 5.4: Approximate training/validation/test splits of the TSM dataset by language.

The hyperparameters used for our experiments are listed in Table 5.5. As optimiser, we have used Adafactor[5] with learning rate set to 0.0001. The models have been trained on a single NVIDIA A40 GPU with 48 GB RAM.

During training, we have bucketed examples with similar length to form training batches, and have passed the batches into the dataloader in random order.

[5]We set the following arguments: scale_parameter=False, relative_step=False, warmup_init=False)

To select the values for the α parameters, we have first identified plausible ranges by initially performing a random search over the parameter space. We have then tuned the parameters manually, using subjective judgement based on our assessment of the quality of the generated text and the validation set performance. We have repeated the process for each dataset, as it is unlikely that a single set of parameters would be ideal for all. The β coefficients have also been set manually, through an iterative process of examination of preliminary results and generated text samples, adjustment, and repetition, and a subjective examination of what constitutes an "acceptable" minimum threshold. The γ constants in Section 5.3.5, which determine when to stop training, were also set in a similar way. The τ temperature parameter for the Gumbel-softmax sampling was set to 1.1 and we have not experimented with other values.

5.8 Limitations

One limitation of the proposed approach is that the experiments have only been conducted on datasets consisting of relatively short examples. However, since longer texts can always be subdivided, this limitation may not be very significant. A second limitation is that the approach has only been tested on five languages, and we do not know how it would perform in more diverse languages, such as Japanese or Chinese. A third limitation is that the method requires use of a GPU with a substantial amount of memory, in order to be able to store all the component models during training. The use of parameter-efficient fine-tuning methods, such as LoRA (Hu et al., 2022) may be able to mitigate this issue.

Hyperparameter	**Value**
General	
Optimisation algorithm	Adafactor
Learning rate	1×10^{-5}
Weight decay	0
Batch size (train)	5
Max original length	32
Precision	fp32
Coefficients (Equations 5.1 and 5.3)	*MARC — TSM dataset*
Victim (α_v)	15 — 15
Similarity (α_s)	20 — 25
Language consistency (α_l)	20 — 20
KL (α_{KL})	1 — 4
Diversity (α_d)	0.1 — 10
Victim threshold (β_v)	$1 - 1/c$
Similarity threshold (β_s)	0.75
Language consistency threshold (β_l)	0.0
KL threshold (β_{KL})	2
Gumbel-softmax temperature τ	1.1
# Gumbel samples	5
Early-stopping criteria (Section 5.3.5)	
Victim score (γ_v)	7
Similarity score (γ_s)	0.75
Language consistency score (γ_l)	1
KL score (γ_{KL})	0.6
Validation frequency	Every 24 batches
Patience	12
Test-set generation	
Batch size (eval)	5
# generated sequences (n)	32
# beams	32
# beam groups	16
Diversity penalty	1
Top-p	0.98
Temperature	1
Min generated length	$\max(0, l - 2 - \text{floor}(l/4))$
Max generated length	$l + 2$

Table 5.5: Hyperparameters. c is the number of classes in the dataset and l is the batch length of generated text during evaluation (after padding).

Chapter 6

Conclusion

In this thesis we have tackled textual adversarial attacks, with the primary aim of demonstrating that the generative approach is a viable alternative to the predominant token-modification methods of the field today. To do so, we have identified and tackled four research questions and units of work, which we will summarise below.

In our first unit of work, we have presented a survey paper that has categorised and unified the components that have made up up token-modification adversarial attacks. We first covered goal functions, which define what the adversary would like to achieve, and allow for a quantifiable measure of progress towards that goal. We next covered the wide range of proposed transformations: the functions that alter one set of ordered tokens into another. After, we covered search methods, which repeatedly apply transformations until the adversarial example is found, or the search fails. Finally, we have also surveyed the diverse range of constraints of the field, which determine what makes successful or unsuccessful adversarial examples. We concluded by proposing future directions for the field, such as more standardised attack scenarios, multi-token attacks, and incorporating human judgements.

In the second unit of work, we have proposed a black-box adversarial attack that uses a reinforcement learning algorithm to fine-tune a seq2seq paraphrase model. We have designed and incorporated a novel reward function that encourages misclassifications in the victim model while penalising deviances from a set of validity constraints. Our experimental results have shown that the fine-tuned model has been able to produce many more adversarial examples than the original paraphrase model, and have also proved much more effective and efficient than a set of compared attacks. Our ablation studies investigated the effect of the decoding method, alternate text classification scenarios, and revealed some strategies that the model learned to create its adversarial attacks. There remains scope to (amongst others) explore more efficient algorithms, exploring different reward functions, or incorporating human judgements.

For our third unit of work, we have presented an approach for creating flexible white-box adversarial attacks against text classifiers. The proposed approach has used an expressive generative model to generate the attacks, and has also incorporated component models into the training objective for textual fluency and semantic consistency. A key component of the approach is the introduction of vocabulary-mapping matrices that remap the vocabularies across all the components, which allows for a training objective that is both

fully-differentiable and highly configurable. Our experimental results across six datasets and two victim models have confirmed the effectiveness of the proposed approach, and additionally our ablation analysis has shown the impact of the key parameters on the attack performance. Future research might aim to integrate other components into the training objective, enhance the vocabulary-mapping matrices, or adapt the approach to other NLP tasks.

In the fourth and final unit of work, we have presented a white-box generative approach for attacking multilingual text classifiers, adapting the training objective from the third unit of work. We have adapted the approach to encourage language adherence in the generated text by incorporating a language detection model into the training objective. Experimental results across two multilingual datasets have confirmed the effectiveness of the proposed approach, particularly in a low number of queries scenario or when richer, sentence-level transforms are desired. We have also presented a qualitative analysis of the key parameters on the characteristics of the generated text, and have highlighted the strengths and weaknesses of the approach. It would be interesting to adapt this framework to better target linguistically diverse languages, such as Russian or Japanese, or to attack models that perform other multilingual NLP tasks, such as those doing cross-lingual classification.

Overall, our most significant achievement has been the development of training approaches for the consistent training of generative models across a number of scenarios. Future work can build on our approaches to further explore the field, particularly since we have made all of our code freely available. Our work also has practical applications beyond adversarial attack generation, of which the most significant is the potential use of the token-mapping matrices proposed in our third and fourth research questions for other NLP tasks. These matrices can be used whenever one has to cascade two or more models with different vocabularies without having to detokenise the output of the earlier models, which retains end-to-end differentiability for the entire pipeline and can benefit many composite NLP tasks. A second practical use is that generative models trained with our approach can be later shared and reused as a generator of hard examples. Because adversarial examples are believed to be transferable across similar victim models, such hard examples could be used to quickly increase the robustness of similar systems. Finally, the work may later prove to have commercial applications, although they are not obvious at present.

Although I am happy with my thesis, if I had my time again I would make a few changes. Firstly, I would choose to not do a survey paper, which was very time-consuming and required a lot of updating. Moreover, it was difficult to include useful information such as the performance of components, partly because each paper has used different threat models, evaluation metrics and so forth. Second, the attempt to align reward functions with human expectations for adversarial examples was ultimately frustrating. I discovered the hard way that automated metrics have limitations, and that human judgement is difficult to emulate and integrate effectively. Today, I would probably attempt to directly incorporate human judgements into the training approach, recognising that automated metrics and I will not always agree on the acceptability of a generated adversarial example. Additionally, I would have liked to conduct human studies for the two white-box papers, as we did in the black-box study. However, since human studies are often undervalued in the field and are costly and time-intensive, we chose not to include them. It's unfortunate, as they would have addressed a known limitation in these papers and the results would have been very interesting.

For future work, since the most natural-sounding text is currently achieved by large language models (LLMs), any shift toward these models and away from established token-modification approaches would be valuable. LLMs are becoming widespread in industry (e.g., retrieval-augmented generation (RAG), agents, coding assistants, multimodal systems), making it important to explore adversarial attacks across more diverse NLP tasks. Additionally, more research on multilingual and cross-lingual attacks, which remain significantly underexplored, would be valuable.

To summarise, the main motivation of the thesis has been to develop generative approaches to creating adversarial examples, and we have achieved this to a satisfactory level. Although much more work remains to be done in this area, this thesis has proved that the generative line of research is viable and has the potential to quickly produce diverse, high-quality adversarial examples, which can be used for both model attack and defense. Finally, although certainly hard at times, I have enjoyed my PhD journey and am very grateful and thankful for the experience.

References

Basemah Alshemali and Jugal Kalita. 2019. Adversarial examples in arabic. In *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 371–376.

Basemah Alshemali and Jugal Kalita. 2020. Improving the Reliability of Deep Neural Networks in NLP: A Review. *Knowledge-Based Systems*, 191:105210.

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo Jhang Ho, Mani B. Srivastava, and Kai Wei Chang. 2020. Generating Natural Language Adversarial Examples. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018*, pages 2890–2896.

Wissam Antoun, Fady Baly, and Hazem Hajj. 2020. Arabert: Transformer-based model for arabic language understanding. In *LREC 2020 Workshop Language Resources and Evaluation Conference 11–16 May 2020*, page 9.

Francesco Barbieri, Luis Espinosa Anke, and Jose Camacho-Collados. 2022. XLM-T: Multilingual language models in Twitter for sentiment analysis and beyond. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 258–266, Marseille, France. European Language Resources Association.

Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and Natural Noise Both Break Neural Machine Translation. In *6th International Conference on Learning Representations, {ICLR} 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer.

Nathaniel Berger, Stefan Riezler, Sebastian Ebert, and Artem Sokolov. 2021. Don't search for a search method - simple heuristics suffice for adversarial text attacks. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 8216–8224. Association for Computational Linguistics.

Battista Biggio and Fabio Roli. 2018. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331.

Nicholas Boucher, Ilia Shumailov, Ross Anderson, and Nicolas Papernot. 2022. Bad characters: Imperceptible NLP attacks. In *43rd IEEE Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, May 22-26, 2022*, pages 1987–2004. IEEE.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Oana-Maria Camburu, Brendan Shillingford, Pasquale Minervini, Thomas Lukasiewicz, and Phil Blunsom. 2020. Make Up Your Mind! Adversarial Generation of Inconsistent Natural Language Explanations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4157–4165, Online. Association for Computational Linguistics.

Nicholas Carlini and David Wagner. 2017. Towards Evaluating the Robustness of Neural Networks. In *Proceedings - IEEE Symposium on Security and Privacy*, pages 39–57. IEEE.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder for English. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174, Brussels, Belgium. Association for Computational Linguistics.

Guoqin Chang, Haichang Gao, Zhou Yao, and Haoquan Xiong. 2023. Textguise: Adaptive adversarial example attacks on text classification model. *Neurocomputing*, 529:190–203.

Moses S. Charikar. 2002. Similarity estimation techniques from rounding algorithms. In *Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing*, STOC '02, page 380–388, New York, NY, USA. Association for Computing Machinery.

Qidong Chen, Jun Sun, and Vasile Palade. 2023. A word-level adversarial attack method based on sememes and an improved quantum-behaved particle swarm optimization. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–12.

Minhao Cheng, Jinfeng Yi, Pin-Yu Chen, Huan Zhang, and Cho-Jui Hsieh. 2020. Seq2Sick: Evaluating the Robustness of Sequence-to-Sequence Models with Adversarial Examples. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):3601–3608.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *ICLR*.

Prithiviraj Damodaran. 2021. Parrot: Paraphrase generation for nlu.

Kareem Darwish and Hamdy Mubarak. 2016. Farasa: A new fast and accurate Arabic word segmenter. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1070–1074, Portorož, Slovenia. European Language Resources Association (ELRA).

Sourya Dipta Das, Ayan Basak, Soumil Mandal, and Dipankar Das. 2021. Advcodemix: Adversarial attack on code-mixed data. *5th Joint International Conference on Data Science & Management of Data (9th ACM IKDD CODS and 27th COMAD)*.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Proceedings of the 11th International AAAI Conference on Web and Social Media*, ICWSM '17, pages 512–515.

Daniel Deutsch, Rotem Dror, and Dan Roth. 2022. On the limitations of reference-free evaluations of generated text. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 10960–10977, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Antreas Dionysiou and Elias Athanasopoulos. 2021. Unicode evil: Evading nlp systems using visual similarities of text characters. In *Proceedings of the 14th ACM Workshop on Artificial Intelligence and Security*, AISec '21, page 1–12, New York, NY, USA. Association for Computing Machinery.

Zhendong Dong, Qiang Dong, and Changling Hao. 2010. HowNet and Its Computation of Meaning. In *Coling 2010: Demonstrations*, pages 53–56, Beijing, China. Coling 2010 Organizing Committee.

Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. 2018. The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1383–1392. Association for Computational Linguistics.

Javid Ebrahimi, Daniel Lowd, and Dejing Dou. 2018a. On Adversarial Examples for Character-Level Neural Machine Translation. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 653–663, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

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

Steffen Eger and Yannik Benz. 2020. From hero to zéroe: A benchmark of low-level adversarial attacks. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing, AACL/IJCNLP 2020, Suzhou, China, December 4-7, 2020*, pages 786–803. Association for Computational Linguistics.

Steffen Eger, Gözde Gül İşgüder, Andreas Rücklé, Ji-Ung Lee, Claudia Schulz, Mohsen Mesgar, Krishnkant Swarnkar, Edwin Simpson, and Iryna Gurevych. 2019. Text Processing Like Humans Do: Visually Attacking and Shielding NLP Systems. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1634–1647, Minneapolis, Minnesota. Association for Computational Linguistics.

Denis Emelin, Ivan Titov, and Rico Sennrich. 2020. Detecting word sense disambiguation biases in machine translation for model-agnostic adversarial attacks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 7635–7653. Association for Computational Linguistics.

Chris Emmery, Ákos Kádár, and Grzegorz Chrupala. 2021. Adversarial stylometry in the wild: Transferable lexical substitution attacks on author profiling. In *EACL*, pages 2388–2402.

Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. 2018. Robust physical-world attacks on deep learning visual classification. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 1625–1634. Computer Vision Foundation / IEEE Computer Society.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. Bradford Books.

Brian Formento, Chuan Sheng Foo, Luu Anh Tuan, and See Kiong Ng. 2023. Using punctuation as an adversarial attack on deep learning-based NLP systems: An empirical study. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1–34, Dubrovnik, Croatia. Association for Computational Linguistics.

I. Fursov, Alexey Zaytsev, Pavel Burnyshev, Ekaterina Dmitrieva, Nikita Klyuchnikov, Andrey Kravchenko, E. Artemova, and Evgeny Burnaev. 2021. A differentiable language model adversarial attack on text classifiers. *IEEE Access*, PP:1–1.

Piotr Gaiński and Klaudia Bałazy. 2023. Step by step loss goes very far: Multi-step quantization for adversarial text attacks. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2038–2048, Dubrovnik, Croatia. Association for Computational Linguistics.

Wee Chung Gan and Hwee Tou Ng. 2019. Improving the robustness of question answering systems to question paraphrasing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6065–6075, Florence, Italy. Association for Computational Linguistics.

Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. 2018. Black-Box Generation of Adversarial Text Sequences to Evade Deep Learning Classifiers. *Proceedings - 2018 IEEE Symposium on Security and Privacy Workshops, SPW 2018*, pages 50–56.

Siddhant Garg and Goutham Ramakrishnan. 2020. BAE: BERT-based Adversarial Examples for Text Classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6174–6181, Online. Association for Computational Linguistics.

Justin Gilmer, Ryan P. Adams, Ian J Goodfellow, David Andersen, and George E. Dahl. 2018. Motivating the Rules of the Game for Adversarial Example Research. *CoRR*, abs/1807.0.

Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. In *International Conference on Learning Representations*.

Will Grathwohl, Dami Choi, Yuhuai Wu, Geoffrey Roeder, and David Duvenaud. 2018. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. In *Proceedings of the 6th International Conference on Learning Representations, ICLR 2018*.

Chuan Guo, Alexandre Sablayrolles, Hervé Jégou, and Douwe Kiela. 2021. Gradient-based adversarial attacks against text transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5747–5757, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xiangzhe Guo, Shikui Tu, and Lei Xu. 2022. Learning to generate textual adversarial examples. In *Artificial Neural Networks and Machine Learning - ICANN 2022 - 31st International Conference on Artificial Neural Networks, Bristol, UK, September 6-9, 2022, Proceedings, Part I*, volume 13529 of *Lecture Notes in Computer Science*, pages 195–206. Springer.

Wenjuan Han, Liwen Zhang, Yong Jiang, and Kewei Tu. 2020. Adversarial attack and defense of structured prediction models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2327–2338, Online. Association for Computational Linguistics.

David Harbecke and Christoph Alt. 2020. Considering likelihood in NLP classification explanations with occlusion and language modeling. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop, ACL 2020, Online, July 5-10, 2020*, pages 111–117. Association for Computational Linguistics.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. 2018. Deep reinforcement learning that matters. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press.

David Herel, Hugo Cisneros, and Tomas Mikolov. 2022. Preserving semantics in textual adversarial attacks. *arXiv preprint arXiv:2211.04205*.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Yu-Lun Hsieh, Minhao Cheng, Da-Cheng Juan, Wei Wei, Wen-Lian Hsu, and Cho-Jui Hsieh. 2019. On the Robustness of Self-Attentive Models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1520–1529, Florence, Italy. Association for Computational Linguistics.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Toward controlled generation of text. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1587–1596. PMLR.

Daphne Ippolito, Daniel Duckworth, Chris Callison-Burch, and Douglas Eck. 2020. Automatic Detection of Generated Text is Easiest when Humans are Fooled. In *ACL*, pages 1808–1822.

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

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Natasha Jaques, Shixiang Gu, Dzmitry Bahdanau, José Miguel Hernández-Lobato, Richard E. Turner, and Douglas Eck. 2017. Sequence tutor: Conservative fine-tuning of sequence generation models with kl-control. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70 of *Proceedings of Machine Learning Research*, pages 1645–1654. PMLR.

Robin Jia and Percy Liang. 2017. Adversarial Examples for Evaluating Reading Comprehension Systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, Copenhagen, Denmark. Association for Computational Linguistics.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is BERT Really Robust? A Strong Baseline for Natural Language Attack on Text Classification and Entailment. *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020*, pages 8018–8025.

Phillip Keung, Yichao Lu, György Szarvas, and Noah A. Smith. 2020. The multilingual amazon reviews corpus. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*.

Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. 2022. Transformers in vision: A survey. *ACM Comput. Surv.*, 54(10s).

Bhargav Kuchipudi, Ravi Teja Nannapaneni, and Qi Liao. 2020. Adversarial machine learning for spam filters. In *Proceedings of the 15th International Conference on Availability, Reliability and Security*, ARES '20, New York, NY, USA. Association for Computing Machinery.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Solomon Kullback and R. A. Leibler. 1951. On information and sufficiency. *Annals of Mathematical Statistics*, 22:79–86.

Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. 2017. Adversarial Machine Learning at Scale. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *Advances in Neural Information Processing Systems (NeurIPS)*.

T. Le, S. Wang, and D. Lee. 2020. Malcom: Generating malicious comments to attack neural fake news detection models. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 282–291, Los Alamitos, CA, USA. IEEE Computer Society.

Thai Le, Jooyoung Lee, Kevin Yen, Yifan Hu, and Dongwon Lee. 2022. Perturbations in the wild: Leveraging human-written text perturbations for realistic adversarial attack and defense. In *Findings of the Association for Computational Linguistics: ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 2953–2965. Association for Computational Linguistics.

Deokjae Lee, Seungyong Moon, Junhyeok Lee, and Hyun Oh Song. 2022. Query-efficient and scalable black-box adversarial attacks on discrete sequential data via Bayesian optimization. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 12478–12497. PMLR.

Yibin Lei, Yu Cao, Dianqi Li, Tianyi Zhou, Meng Fang, and Mykola Pechenizkiy. 2022. Phrase-level textual adversarial attack with label preservation. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1095–1112, Seattle, United States. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual*

Meeting of the Association for Computational Linguistics, pages 7871–7880, Online. Association for Computational Linguistics.

Dianqi Li, Yizhe Zhang, Hao Peng, Liqun Chen, Chris Brockett, Ming-Ting Sun, and Bill Dolan. 2021. Contextualized perturbation for textual adversarial attack. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5053–5069, Online. Association for Computational Linguistics.

Jinfeng Li, Tianyu Du, Shouling Ji, Rong Zhang, Quan Lu, Min Yang, and Ting Wang. 2020a. TextShield: Robust Text Classification Based on Multimodal Embedding and Neural Machine Translation. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 1381–1398. USENIX Association.

Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. 2019. TextBugger: Generating Adversarial Text Against Real-world Applications. In *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*, February. The Internet Society.

Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. *EMNLP 2017 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, pages 2157–2169.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020b. BERT-ATTACK: Adversarial Attack Against BERT Using BERT. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6193–6202, Online. Association for Computational Linguistics.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020c. BERT-ATTACK: Adversarial attack against BERT using BERT. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6193–6202.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi. 2018. Deep Text Classification Can be Fooled. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, {IJCAI-18}*, pages 4208–4215. International Joint Conferences on Artificial Intelligence Organization.

Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Salvatore Candido, and Alexander Rives. 2023. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130.

Hanyu Liu, Chengyuan Cai, and Yanjun Qi. 2023. Expanding scope: Adapting English adversarial attacks to Chinese. In *Proceedings of the 3rd Workshop on Trustworthy Natural Language Processing (TrustNLP 2023)*, pages 276–286, Toronto, Canada. Association for Computational Linguistics.

Shengcai Liu, Ning Lu, Cheng Chen, and Ke Tang. 2022. Efficient combinatorial optimization for word-level adversarial textual attack. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 30:98–111.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742.

Jiaying Lu, Xin Ye, Yi Ren, and Yezhou Yang. 2022. Good, better, best: Textual distractors generation for multi-choice VQA via policy gradient. In *CVPR 2022 Workshop on Open-Domain Retrieval Under a Multi-Modal Setting*, O-DRUM @ CVPR 2022.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards deep learning models resistant to adversarial attacks. *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, pages 1–28.

Rishabh Maheshwary, Saket Maheshwary, and Vikram Pudi. 2021a. Generating natural language attacks in a hard label black box setting. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh*

Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021, pages 13525–13533. AAAI Press.

Rishabh Maheshwary, Saket Maheshwary, and Vikram Pudi. 2021b. A strong baseline for query efficient attacks in a black box setting. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8396–8409, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Gallil Maimon and Lior Rokach. 2022. A universal adversarial policy for text classifiers. *Neural Networks*, 153:282–291.

Vijit Malik, Ashwani Bhat, and Ashutosh Modi. 2021. Adv-olm: Generating textual adversaries via olm.

Eric Malmi, Aliaksei Severyn, and Sascha Rothe. 2020. Unsupervised text style transfer with padded masked language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8671–8680, Online. Association for Computational Linguistics.

P. Malo, A. Sinha, P. Korhonen, J. Wallenius, and P. Takala. 2014. Good debt or bad debt: Detecting semantic orientations in economic texts. *Journal of the Association for Information Science and Technology*, 65.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland. Association for Computational Linguistics.

Leland McInnes, John Healy, and Steve Astels. 2017. hdbscan: Hierarchical density based clustering. *The Journal of Open Source Software*, 2(11).

Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. 2018. Umap: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29):861.

Zhao Meng and Roger Wattenhofer. 2020. A geometry-inspired attack for generating natural language adversarial examples. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 6679–6689. International Committee on Computational Linguistics.

Paul Michel, Xian Li, Graham Neubig, and Juan Pino. 2019. On Evaluation of Adversarial Perturbations for Sequence-to-Sequence Models. 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 3103–3114, Minneapolis, Minnesota. Association for Computational Linguistics.

Ishani Mondal. 2021. BBAEG: towards bert-based biomedical adversarial example generation for text classification. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 5378–5384. Association for Computational Linguistics.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. Deepfool: A simple and accurate method to fool deep neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2574–2582. IEEE Computer Society.

John Morris, Eli Lifland, Jack Lanchantin, Yangfeng Ji, and Yanjun Qi. 2020a. Reevaluating adversarial examples in natural language. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3829–3839, Online. Association for Computational Linguistics.

John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020b. TextAttack: A framework for adversarial attacks, data augmentation, and adversarial training in NLP. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 119–126, Online. Association for Computational Linguistics.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting Word Vectors to Linguistic Constraints. In *Proceedings of HLT-NAACL*.

Daniel Naber, Technische Fakultät, and Universität Bielefeld. 2003. A Rule-Based Style and Grammar Checker.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 271–278, Barcelona, Spain.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the ACL*.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. 2017. Practical black-box attacks against machine learning. *ASIA CCS 2017 - Proceedings of the 2017 ACM Asia Conference on Computer and Communications Security*, pages 506–519.

Nicolas Papernot, Patrick D McDaniel, Ananthram Swami, and Richard E Harang. 2016. Crafting Adversarial Input Sequences for Recurrent Neural Networks. In *2016 IEEE Military Communications Conference, MILCOM 2016, Baltimore, MD, USA, November 1-3, 2016*, pages 49–54. IEEE.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, page 311–318, USA. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe: Global Vectors for Word Representation. In *In EMNLP*.

Jonas Pfeiffer, Naman Goyal, Xi Lin, Xian Li, James Cross, Sebastian Riedel, and Mikel Artetxe. 2022. Lifting the curse of multilinguality by pre-training modular transformers. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3479–3495, Seattle, United States. Association for Computational Linguistics.

Fanchao Qi, Yangyi Chen, Xurui Zhang, Mukai Li, Zhiyuan Liu, and Maosong Sun. 2021. Mind the style of text! adversarial and backdoor attacks based on text style transfer. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4569–4580, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners.

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

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating Natural Language Adversarial Examples through Probability Weighted Word Saliency. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1085–1097.

Yankun Ren, Jianbin Lin, Siliang Tang, Jun Zhou, Shuang Yang, Yuan Qi, and Xiang Ren. 2020. Generating natural language adversarial examples on a large scale with generative models. *CoRR*, abs/2003.10388.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 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.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of nlp models with checklist. In *Association for Computational Linguistics (ACL)*.

Sara Rosenthal, Mihaela A. Bornea, and Avirup Sil. 2021. Are multilingual bert models robust? a case study on adversarial attacks for multilingual question answering. *ArXiv*, abs/2104.07646.

Tom Roth, Yansong Gao, Alsharif Abuadbba, Surya Nepal, and Wei Liu. 2021. Token-modification adversarial attacks for natural language processing: A survey.

Suranjana Samanta and Sameep Mehta. 2017. Towards Crafting Text Adversarial Samples. *arXiv*, abs/1707.0.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108.

Elvis Saravia, Hsien-Chi Toby Liu, Yen-Hao Huang, Junlin Wu, and Yi-Shin Chen. 2018. CARER: Contextualized affect representations for emotion recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3687–3697, Brussels, Belgium. Association for Computational Linguistics.

Yves Scherrer. 2020. TaPaCo: A corpus of sentential paraphrases for 73 languages. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 6868–6873, Marseille, France. European Language Resources Association.

Zhihong Shao, Zhongqin Wu, and Minlie Huang. 2022. Advexpander: Generating natural language adversarial examples by expanding text. *IEEE ACM Trans. Audio Speech Lang. Process.*, 30:1184–1196.

Sahil Singla, Eric Wallace, Shi Feng, and Soheil Feizi. 2019. Understanding Impacts of High-Order Loss Approximations and Features in Deep Learning Interpretation. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5848–5856. PMLR.

Liwei Song, Xinwei Yu, Hsuan-Tung Peng, and Karthik Narasimhan. 2021. Universal adversarial attacks with natural triggers for text classification. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3724–3733, Online. Association for Computational Linguistics.

Charles Stephenson. 1980. The methodology of historical census record linkage: a user's guide to the soundex. *Journal of Family History*, 5(1):112–115.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. MobileBERT: a compact task-agnostic BERT for resource-limited devices. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2158–2170, Online. Association for Computational Linguistics.

Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, page 1057–1063, Cambridge, MA, USA. MIT Press.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*, pages 1–10.

Samson Tan and Shafiq Joty. 2021. Code-mixing on sesame street: Dawn of the adversarial polyglots. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3596–3616, Online. Association for Computational Linguistics.

Samson Tan, Shafiq Joty, Min-Yen Kan, and Richard Socher. 2020. It's Morphin' Time! Combating Linguistic Discrimination with Inflectional Perturbations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2920–2935, Online. Association for Computational Linguistics.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin,

Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Ashwin Vijayakumar, Michael Cogswell, Ramprasaath Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2018. Diverse beam search for improved description of complex scenes. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).

Ashwin K. Vijayakumar, Michael Cogswell, Ramprasaath R. Selvaraju, Qing Sun, Stefan Lee, David J. Crandall, and Dhruv Batra. 2016. Diverse beam search: Decoding diverse solutions from neural sequence models. *CoRR*, abs/1610.02424.

Prashanth Vijayaraghavan and Deb Roy. 2019. Generating black-box adversarial examples for text classifiers using a deep reinforced model. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 711–726. Springer.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019a. Universal adversarial triggers for attacking and analyzing NLP. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2153–2162, Hong Kong, China. Association for Computational Linguistics.

Eric Wallace, Pedro Rodriguez, Shi Feng, Ikuya Yamada, and Jordan Boyd-Graber. 2019b. Trick Me If You Can: Human-in-the-loop Generation of Adversarial Question Answering Examples. *Transactions of the Association for Computational Linguistics*, 7(0):387–401.

Eric Wallace, Mitchell Stern, and Dawn Song. 2020a. Imitation attacks and defenses for black-box machine translation systems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5531–5546, Online. Association for Computational Linguistics.

Eric Wallace, Jens Tuyls, Junlin Wang, Sanjay Subramanian, Matt Gardner, and Sameer Singh. 2020b. AllenNLP Interpret: A framework for explaining predictions of NLP models. *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, Proceedings of System Demonstrations*, pages 7–12.

Juncheng Wan, Jian Yang, Shuming Ma, Dongdong Zhang, Weinan Zhang, Yong Yu, and Zhoujun Li. 2022. PAEG: Phrase-level adversarial example generation for neural machine translation. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 5085–5097, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Boxin Wang, Hengzhi Pei, Boyuan Pan, Qian Chen, Shuohang Wang, and Bo Li. 2020a. T3: Tree-autoencoder constrained adversarial text generation for targeted attack. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6134–6150, Online. Association for Computational Linguistics.

Boxin Wang, Chejian Xu, Xiangyu Liu, Yu Cheng, and Bo Li. 2022. SemAttack: Natural textual attacks via different semantic spaces. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 176–205, Seattle, United States. Association for Computational Linguistics.

Ke Wang and Xiaojun Wan. 2018. Sentigan: Generating sentimental texts via mixture adversarial networks. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 4446–4452. International Joint Conferences on Artificial Intelligence Organization.

Tianlu Wang, Xuezhi Wang, Yao Qin, Ben Packer, Kang Li, Jilin Chen, Alex Beutel, and Ed Chi. 2020b. CAT-gen: Improving robustness in NLP models via controlled adversarial text generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5141–5146, Online. Association for Computational Linguistics.

Xiaosen Wang, Jin Hao, Yichen Yang, and Kun He. 2021a. Natural language adversarial defense through synonym encoding. In *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intel-*

ligence, UAI 2021, Virtual Event, 27-30 July 2021, volume 161 of *Proceedings of Machine Learning Research*, pages 823–833. AUAI Press.

Yuxuan Wang, Wanxiang Che, Ivan Titov, Shay B. Cohen, Zhilin Lei, and Ting Liu. 2021b. A closer look into the robustness of neural dependency parsers using better adversarial examples. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2344–2354, Online. Association for Computational Linguistics.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2018. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*.

Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun. 2023. Transformers in time series: A survey. In *International Joint Conference on Artificial Intelligence(IJCAI)*.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3–4):229–256.

David S. Wishart, Yannick D. Feunang, An Chi Guo, Elvis J. Lo, Ana Marcu, Jason R. Grant, Tanvir Sajed, Daniel Johnson, Carin Li, Zinat Sayeeda, Nazanin Assempour, Ithayavani Iynkkaran, Yifeng Liu, Adam Maciejewski, Nicola Gale, Alex Wilson, Lucy Chin, Ryan Cummings, Diana Le, Allison Pon, Craig Knox, and Michael Wilson. 2018. Drugbank 5.0: a major update to the drugbank database for 2018. *Nucleic Acids Res.*, 46(Database-Issue):D1074–D1082.

Catherine Wong. 2017. Dancin seq2seq: Fooling text classifiers with adversarial text example generation.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Jincheng Xu and Qingfeng Du. 2020. TextTricker: Loss-Based and Gradient-Based Adversarial Attacks on Text Classification Models. *Eng. Appl. Artif. Intell.*, 92:103641.

Lei Xu, Alfredo Cuesta-Infante, Laure Berti-Equille, and Kalyan Veeramachaneni. 2022. R&R: Metric-guided adversarial sentence generation. In *Findings of the Association for Computational Linguistics: AACL-IJCNLP 2022*, pages 438–452, Online only. Association for Computational Linguistics.

Ying Xu, Xu Zhong, Antonio Jimeno Yepes, and Jey Han Lau. 2021. Grey-box adversarial attack and defence for sentiment classification. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4078–4087, Online. Association for Computational Linguistics.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

Yinfei Yang, Daniel Cer, Amin Ahmad, Mandy Guo, Jax Law, Noah Constant, Gustavo Hernandez Abrego, Steve Yuan, Chris Tar, Yun-hsuan Sung, Brian Strope, and Ray Kurzweil. 2020. Multilingual universal sentence encoder for semantic retrieval. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 87–94, Online. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Jin Yong Yoo, John Morris, Eli Lifland, and Yanjun Qi. 2020. Searching for a search method: Benchmarking search algorithms for generating NLP adversarial examples. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 323–332, Online. Association for Computational Linguistics.

Jin Yong Yoo and Yanjun Qi. 2021. Towards improving adversarial training of NLP models. In *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021*, pages 945–956. Association for Computational Linguistics.

Lifan Yuan, YiChi Zhang, Yangyi Chen, and Wei Wei. 2023. Bridge the gap between CV and NLP! a gradient-based textual adversarial attack framework. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 7132–7146, Toronto, Canada. Association for Computational Linguistics.

Liping Yuan, Xiaoqing Zheng, Yi Zhou, Cho-Jui Hsieh, and Kai-Wei Chang. 2021a. On the transferability of adversarial attacks against neural text classifier. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1612–1625, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Weizhe Yuan, Graham Neubig, and Pengfei Liu. 2021b. Bartscore: Evaluating generated text as text generation. In *Advances in Neural Information Processing Systems*, volume 34, pages 27263–27277. Curran Associates, Inc.

Yuan Zang, Fanchao Qi, Chenghao Yang, Zhiyuan Liu, Meng Zhang, Qun Liu, and Maosong Sun. 2020. Word-level Textual Adversarial Attacking as Combinatorial Optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6066–6080, Online. Association for Computational Linguistics.

Guoyang Zeng, Fanchao Qi, Qianrui Zhou, Tingji Zhang, Bairu Hou, Yuan Zang, Zhiyuan Liu, and Maosong Sun. 2021. Openattack: An open-source textual adversarial attack toolkit. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 363–371.

Zhiyuan Zeng and Deyi Xiong. 2021. An empirical study on adversarial attack on NMT: Languages and positions matter. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 454–460, Online. Association for Computational Linguistics.

Pengwei Zhan, Chao Zheng, Jing Yang, Yuxiang Wang, Liming Wang, Yang Wu, and Yunjian Zhang. 2022. PARSE: an efficient search method for black-box adversarial text attacks. In *Proceedings of the 29th International Conference on Computational Linguistics, COLING 2022, Gyeongju, Republic of Korea, October 12-17, 2022*, pages 4776–4787. International Committee on Computational Linguistics.

Huangzhao Zhang, Hao Zhou, Ning Miao, and Lei Li. 2019. Generating fluent adversarial examples for natural languages. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5564–5569, Florence, Italy. Association for Computational Linguistics.

Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Wei Emma Zhang, Quan Z. Sheng, Ahoud Alhazmi, and Chenliang Li. 2020. Adversarial Attacks on Deep-learning Models in Natural Language Processing. *ACM Transactions on Intelligent Systems and Technology*, 11(3):1–41.

Xinze Zhang, Junzhe Zhang, Zhenhua Chen, and Kun He. 2021. Crafting adversarial examples for neural machine translation. 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 1967–1977, Online. Association for Computational Linguistics.

Zhengli Zhao, Dheeru Dua, and Sameer Singh. 2018. Generating natural adversarial examples. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Wangchunshu Zhou, Tao Ge, Ke Xu, Furu Wei, and Ming Zhou. 2019. Bert-based lexical substitution. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 3368–3373. Association for Computational Linguistics.

Bin Zhu, Zhaoquan Gu, Yaguan Qian, Francis Lau, and Zhihong Tian. 2022. Leveraging transferability and improved beam search in textual adversarial attacks. *Neurocomputing*, 500:135–142.

Hai Zhu, Qinyang Zhao, and Yuren Wu. 2023. Beamattack: Generating high-quality textual adversarial examples through beam search and mixed semantic spaces. In *Advances in Knowledge Discovery and Data Mining: 27th Pacific-Asia Conference on Knowledge Discovery and Data Mining, PAKDD 2023, Osaka, Japan, May 25–28, 2023, Proceedings, Part II*, page 454–465, Berlin, Heidelberg. Springer-Verlag.

Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2019. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*.