Evaluating the Robustness of Deep learning Models for NLP

Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science
in
Computer Science and Engineering by Research

by

Rishabh Maheshwary
2019701023
`rishabh.maheshwary@research.iiit.ac.in`

International Institute of Information Technology
(Deemed to be University)
Hyderabad - 500 032, INDIA
March 2023

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "**Evaluating the Robustness of Deep learning Models for NLP**" by Rishabh Maheshwary, has been carried out under my supervision and is not submitted elsewhere for a degree.

Date	Adviser: Dr. Vikram Pudi

To my dearest parents for their

endless love and support.

Acknowledgments

I would like to express my heartfelt gratitude towards IIIT Hyderabad community at large. My entire journey here has been wonderful and perhaps the most memorable experience of all.

This thesis would not be possible without the guidance of Prof. Vikram Pudi. I am deeply grateful for his guidance and support throughout my research. He provided me with ample support and complete autonomy to work on the problems that I was interested in, and he consistently encouraged me to continue pushing forward even when the results were not immediately clear. Despite his busy schedule, he always made time to meet with me to discuss the progress of my work and potential future directions. I am also thankful to him for allowing me to collaborate with others and for giving me the freedom to work at my own pace. His support and guidance were invaluable to me.

I am deeply grateful to my brother Saket for his support and guidance throughout my research. He played a crucial role in helping me to brainstorm potential approaches during the initial stages of my research, and his advice, interactions, and valuable insights were instrumental in getting my research published. I am thankful for his unwavering support and guidance.

I would like to thank Vivek in particular for his valuable input and advice. He took the time to carefully analyze my research problem and provide me with valuable insights.

I am grateful to IIIT Hyderabad as an institution for providing me with an outstanding atmosphere for conducting my research. The courses, assignments, and seminar talks offered numerous opportunities for students to pursue their research interests. In today's world, access to adequate computational infrastructure is crucial for conducting research in computer science, especially when dealing with large data sets. I am incredibly fortunate to have had access to a wealth of computational resources at my disposal, which enabled me to carry out my experiments successfully. I am deeply indebted to the institute for providing me with such a conducive environment for conducting my research.

I am grateful to my family for their continued support and sacrifices, without which I would not have been able to complete my masters on time. I am particularly thankful to my parents for allowing me to chart my own career path. I am lucky to have a large network of friends, ranging from juniors to seniors, from B.Techs to PhDs, who made my time at IIIT truly memorable.

Abstract

The significance of deep neural networks (DNNs) has been well established through its success in a variety of tasks. However, recent studies have shown that DNNs are vulnerable to *adversarial examples* — inputs crafted by adding small perturbations to the original input. Such perturbations are almost imperceptible to humans but deceive DNNs thus raising major concerns about their utility in real world applications. Although, existing adversarial attack methods in NLP have achieved high success rate in attacking DNNs, they either require detailed information about the target model, training data or need a large amount of queries to generate attacks. Therefore, such attack methods are not realistic as it does not reflect the types of attacks that can be encountered in the real world and are less effective as attacks relying on model information and excessive queries can be easily defended against. In this thesis, we address the above mentioned drawbacks by proposing two realistic attack settings — *hard label black box setting* and *limited query setting*. Next, we propose two novel attack methods that crafts plausible and semantically similar adversarial examples in the above settings. The first attack method uses a population based optimization procedure to craft adversarial examples in the hard label black box setting. The second method is a query efficient method that leverages word attention scores and locality sensitive hashing to find important words for substitution in the limited query setting. We benchmark our results across the same search space used in prior attacks so as to ensure fair and consistent comparison. To improve the quality of generated adversarial examples, we propose an alternative method that uses masked language model to find candidate words for substitution by considering the information of both the original word and its surrounding context. We demonstrate the efficacy of each of our proposed approach by attacking NLP models for text classification and natural language inference task. In addition to that we use adversarial examples to evaluate the robustness and generalization of recent math word problem solvers. Our results showcase that DNNs for the above tasks are not robust as they can be deceived by our proposed attack methods in a highly restricted setting. We conduct human evaluation studies to verify the validity and quality of generated adversarial examples.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the past decade, the significance of deep neural networks (DNNs) over traditional machine learning models [1, 2, 3, 4, 5, 6, 7, 8, 9] has been well established through its success on a variety of tasks such as classification [10, 11, 12, 13, 14, 15, 16], question answering [17, 18, 19, 20], machine translation [21, 22], object recognition [23, 24], speech [25, 26], image generation [27, 28], self driving cars [29], etc. Despite its tremendous popularity and wide use, the vulnerabilities of DNNs are still largely unknown, which raises concerns about their use in mission-critical applications where a single mistake can have serious consequences. To build trust and confidence in DNNs, it is necessary to study their robustness in real-world scenarios so as to gain a better understanding of their limitations, vulnerabilities, and performance under different conditions.

Recent studies [30, 31, 32] have demonstrated that DNNs are not robust as they are vulnerable to *adversarial examples*. Adversarial examples are strategically crafted inputs to fool DNNs to a mistake. Such inputs are crafted by adding small perturbations to the original input such that modified input is semantically similar to the original input but is mis-classified by the target model. The procedure used to craft such inputs is called *adversarial attack* method. Prior work related to vision and speech [30, 31, 32, 33, 34, 35] have explored a variety of adversarial attack methods, however generating adversarial attacks in text is still a challenging task because of (1) discrete nature of text as replacing a single word in text can completely alter its semantics and (2) grammatical correctness and fluency. Due to this, attack methods proposed in vision and speech are not directly applicable in the text domain. Initial works in NLP [36, 37, 38, 39] crafted adversarial inputs by introducing perturbations at character level. Later on, [40, 41, 42, 43, 44, 45, 46] introduced word level substitutions to attack text classification and entailment models. Although the above approaches have achieved high attack success rate they have several major drawbacks.

Full access to model or confidence scores — Methods proposed earlier [36, 37, 38, 41, 40, 42, 43, 44, 45, 46] either require access to the target model during testing or use the model's confidence scores to generate adversarial attacks. Some other works [47, 48] use GANs or rely on training substitute models. All the above methods are not effective in a realistic setting where the attacker does not have access to the model's architecture, parameters, confidence scores or training data. Crafting adversarial attacks with no access to the model or confidence scores is more realistic and relevant, as it simulates the type of attacks that can be carried out in the real world. Additionally, this approach provides insight into a model's vulnerabilities and help improve its robustness.

Query inefficient — None of the prior approaches consider the amount of queries required to generate attacks. This lowers the practicality, effectiveness, and stealthiness of the attack when applied to real-world systems where there is a cost associated with each query. For example, if an adversarial attack method requires large number of queries to generate an attack, it is not practical to use the method in a real-world where the cost of conducting the attack is high. Additionally, if the number of queries required is too large, the attack will be less effective, as the model may be able to learn and adapt to the attack over time. Finally, a large number of queries will make the attack more noticeable, which can reduce its stealthiness. Combinatorial optimization based attacks [40, 44] are extremely slow and require massive amount of queries to generate adversarial examples. Greedy attack methods [42, 43, 41] uses ranking mechanisms that are inefficient on input of larger lengths.

Grammatically incorrect and non-fluent examples — Prior attacks find adversarial examples by replacing important words with synonyms generated either using a lexical database such as WordNet [49] or from the nearest neighbour in the counter-fitted embedding [50] space. However, such methods takes only the word level similarity to find synonyms but does not consider the overall context surrounding the replaced word thus generating out of context synonyms. Moreover, it yields unnatural and complex replacements which results in grammatically incorrect and non-fluent adversarial examples.

Inconsistent search space — [51] demonstrated that attacks in [52, 45, 46] do not maintain a consistent search space (i.e. the source used to get synonyms for a word) while comparing various attacks. This does not ensure a fair comparison among attack methods as it is hard to distinguish whether the increase in attack success rate is due to the improved attack method or modified search space.

1.1 Contribution

In this thesis, we address the above mentioned drawbacks by introducing two different realistic attack setting to evaluate NLP model — *hard label black box setting* and *limited query setting*. Next we propose two novel attack methods that generate plausible and semantically similar adversarial examples in the above attack settings. Apart from this, we evaluate the robustness and generalization of math word problem solvers using adversarial examples. We also conducted human evaluation to ensure that the generated adversarial examples are valid, semantically similar and grammatically correct.

- **Hard Label Black Box Setting** — First, we propose a novel attack setting called the *hard label black box setting*. This attack setting does not requires information about the model architecture, parameters, confidence scores or training data. It only assumes access to the top predicted label. Next, we propose a mechanism that uses population-based optimization procedure to successfully generate adversarial examples in the hard label black box setting, where all the prior methods fail.

In comparison to previous attack strategies, our attack achieves a higher success rate (more than 90%) and lower perturbation rate that too in a highly restricted setting. The hard label black box setting has been explored recently in vision [53, 54] but to the best of our knowledge we are first to explore it in NLP domain.

- **Limited Query Setting** — First, we propose a *novel ranking mechanism* that jointly leveraging word attention scores and locality sensitivity hashing (LSH) to rank the input words. It takes significantly less number of queries and works best in a *limited query setting* when compared to prior black box methods. Next, following [51] we benchmark our attack method on the same search space used in the respective baselines. Further, we evaluate its effectiveness by comparing it with *four* baselines across *three* different search spaces. On an average, our method is 50% faster as it takes 75% lesser queries than the prior attacks while compromising the attack success rate by less than 2%.

- **Context aware adversarial attack** — We propose an attack that generates candidate words using the influence of both the original word (to be replaced) as well as its surrounding context. It uses BERT [19] (a Masked Language Model (MLM) trained for masked word and next sentence prediction task) to generate candidates for each word to be replaced in the input.

- **Evaluating Math Word Problem Solvers** — Although adversarial examples are commonly used for various NLP tasks such as question answering [55, 56, 57], machine translation [58, 59], dialogue systems [60], text classification [40, 42, 52, 61, 62], natural language inference [63], etc there has been no work that uses adversarial examples to evaluate Math Word Problem solvers. A Math Word Problem (MWP) consists of a natural language text which describes a world state involving some known and unknown quantities. The task is to parse the text and generate equations that can help find the value of unknown quantities. An example is given in Table 6.1. Standard accuracy metrics have shown that Math Word Problem (MWP) solvers have achieved high performance on benchmark datasets. However, the extent to which existing MWP solvers truly understand language and its relation with numbers is still unclear. In chapter 6, we bridge this gap and evaluate the robustness of state-of-the-art MWP solvers against adversarial examples. Generating adversarial attacks for MWP is a challenging task as apart from preserving textual semantics, numerical value also needs to be preserved. The text should make mathematical sense, and the sequence of events must be maintained such that humans generate the same equations from the problem text. We propose two methods *Question Reordering* and *Sentence Paraphrasing* to generate adversarial examples on three MWP solvers across two benchmark datasets. On average, the generated adversarial examples are able to reduce the accuracy of MWP solvers by over 40%. Further, we experiment with different type of input embeddings and perform adversarial training using our proposed methods.

1.2 Thesis structure explained

The thesis is organized as follows. In chapter 2 we discuss the existing adversarial attack methods for various NLP tasks in detail. The methods are categorized into *white box attacks*, *score based attacks*, *transfer based attacks* and *decision based attacks* depending upon the amount of access of the target model required. In chapter 3 we present the hard label black box setting and the population based optimization algorithm used to generate attacks in such a setting. In chapter 4 we introduce the limited query setting and the novel ranking mechanism that leverages attention and LSHfor ranking. In chapter 5 we introduce an attack method that generates high quality adversarial examples using MLMs. In chapter 6 we evaluate the generalization and robustness of MWP solvers against adversarial examples. Finally, we conclude by summarizing the contributions made in the thesis and potential future work.

Chapter 2

Related Work

2.1 Adversarial Examples

Studies on adversarial examples in the vison are more active than in NLP. Most of the earlier works in the computer vision generated adversarial examples by adding imperceptible perturbations to the input image. [30] was the first work to attack deep neural image classifiers. It adds a perturbation to the image found by minimizing the loss function of the network and the distance between the original and perturbed image. [31] proposed Fast Gradient Sign Method (FGSM) to generate adversarial examples. FGSM method adds perturbations computed using the sign of the gradient of the cost function with respect to input. [31] also argues that adversarial examples results due to linear decision boundry of DNNs. Inspired by the FGSM, [34] proposed DeepFool that applies FGSM iteratively to compute smaller perturbation. [64] treats DNN as a black box and trains a substitute models to mimic the decision boundary of the target model. Adversarial examples are generated against the substitute models and then are transferred to target models. [32] proposed Jacobian based Saliency Map Attack (JSMA) that uses the uses the gradient of the loss function with respect to input image to computes a saliency map of pixels. Given the saliency map, it picks the most important pixel and modify it to increase the likelihood of the target class. This process is repeated till the image is mis-classified. [33] constructed three different attacks based upon various distance metric and various choices of loss function to demonstrate that DNNs trained with defensive distillation [65] are not robust.

[53] was the first work to attack DNNs image classifier by observing only the top predicted label also known as hard label black box setting. It proposed boundary attack that initializes the attack algorithm from a point that is already adversarial and then performs a random walk along the boundary between the adversarial and the non-adversarial region. Later, [54] used zero order optimization to reduce the number of queries in hard label setting. [66] proposed three realistic threat models – limited query setting, hard label setting and partial information setting to evaluate the robustness of DNNs.

2.2 Adversarial Examples in NLP

White-box attacks: Such attack strategies rely on the gradient information of the loss with respect to input to generate an attack. Inspired by the fast gradient sign method (FGSM) in the image domain, attackers generate adversarial examples by calculating the gradient of text vectors in a model. HotFlip [36] generates perturbations at a character level. It flips characters in the input that maximizes the loss of the target model. Although, this method is efficient it results in misspellings that highly alters the semantics of the input. Following this, [39] used gradient information to find important positions in the input text and introduces character level perturbations (insertion, deletion and replacement) on those positions to generate adversarial examples. On similar lines, [41] proposed a word level attack method. It first find important words by computing the gradient of loss function with respect to each word in the input and replaces those words with similar words. Inspired by this, [56] does a gradient-guided search over words to find short trigger sequences. These triggers when concatenated with the input forces the model to generate incorrect predictions. Similarly, [67] leveraged adversarially regularized autoencoder (ARAE) [68] to generate triggers. Although this method performs well, the generated trigger sequences are random tokens from the vocabulary, thus resulting in meaningless sentences. In contrast to the above attacks [69] introduced adversarial perturbations by restricting the directions of the perturbations toward the locations of existing words in the word embedding space. Recently, [70] attacked transformer [22] models by searching for a distribution of adversarial examples parameterized by a continuous-valued matrix. Furthermore, they benchmarked their attack in a black box setting by transferring it across multiple models. All the above attacks require access to the detailed model information, which is not available in real world scenarios.

Score-based attacks: This category of attack requires access to the target models confidence scores or class probabilities to craft adversarial inputs. Most score-based attacks generate adversarial examples by first finding important words which highly impact the confidence scores of the target class and then replaces those words with similar words. The replacements are done till the model mis-classifies the input. At first, [38] introduced DeepwordBug which generates character level perturbations on the important words in the input. Later, [71] used Markov chain Monte Carlo sampling approach to generate adversarial inputs. Then [42] used saliency based word ranking to find important words and replaced those with synonyms from WordNet [49]. On similar lines, [43] proposed TextFooler which substitutes important word in the input with synonyms from [50]. Recently [45, 46] used a masked language models (MLMs) to find substitutions for important words in the input text. On similar lines, [72] used MLMs to propose three contextualized perturbations replace, insert and merge, that results in perturbed inputs of varied lengths. [73] proposed Controlled Adversarial Text Generation (CATGen) model that, given an input text, generates adversarial texts through controllable attributes that are known to be irrelevant to task labels. It uses an encoder, decoder architecture with an attriute classifier. [57] proposed T3 which used tree autoencoders to generate adversarial sentences in a non-monotonic order [74] for classification and question answering systems. Unlike the above strategies, the work in [40, 44] use combinatorial optimization algorithms to craft adversarial inputs. Specifially, [40] uses Genetic Algorithm (GA) to find plausible adversarial examples in a given search space. [44] improves upon it by using Particle Swarm Optimization (PSO) and uses HowNet [75] to find word substitutions. In such algorithms, the target label prediction probability is used as an optimization criteria at each iteration of the optimization algorithm.

Transfer-based attacks: Transfer-based attacks train a substitute model by repeatedly querying the target model so as to mimic the decison boundary of the target classifier. Adversarial examples are then generated against the substitute models and are transferred to the target models. Such attacks rely on information about the training data on which the target models are trained and relies on the assumption that adversarial examples successfully transfer between models of different architectures. Motivated by this [48] used substitute models and reinforcement learning methods to attack target models. [76, 77, 78] did an extensive study to systematically evaluate transferability of adversarial examples for text classification models and explore how various factors, including network architecture, tokenization scheme, word embedding, and model capacity, affect the transferability. Transfer-based attacks are expensive to apply as they require training a substitute model.

Decision-based attacks: Decision-based attacks only depends on the final predicted label of the target classifier. Most of the existing decision based attack methods in NLP are for question answering, machine translation and paraphrasing tasks. [55] was the initial works on adversarial attacks in NLP domain. They proposed two techniques AddSent and Addany that adds distractor sentences to a passage which are un-related to the passage context thus forcing the model to generate incorrect answers. [79] used paraphrasing on the question text to evaluate and improve the robustness of QA systems. [80] used back translation and syntactic templates to generate syntactically controlled paraphrases to attack models. [81] proposed a set of rules SEARs to test the robustness and generalisability of NLP models across multiple tasks. [82, 63] proposed an NLI test set that substitutes words with phrases requiring lexical and world knowledge. [83] used reinforcement learning to generate adversarial examples for machine translation models. The only relevant prior decision-based attack for classification task [47] uses Generative Adversarial Network (GANs) which are hard to train and require access to training data.

2.3 Math Word problem Solvers

Math Word Solvers: Many research efforts have been undertaken in the recent past to solve the challenging Math Word Problem (MWP) task. Broadly, these solvers can be categorized into statistical learning based and deep learning based models. Traditional approaches focused more on statistical machine learning [6, 5] with the aim of categorizing equations into templates and extracting key patterns in the problem text. Recently, due to the advent of deep learning in NLP, solvers have witnessed a considerable increase in their performances. [84] modelled MWP task as a sequence to sequence task and used LSTM's [85] for learning problem representations. [86] focused on learning representations for operators and operands. [87, 88] used tree structures for decoding process. [89] modelled question as a graph to map quantities and their attributes.

Datasets: Existing datasets which have been used as benchmark for english language includes MaWPS [90] and Chinese language dataset Math23K [84]. These datasets although constrained by their size deal with algebraic problems of similar difficulty level. Recently, ASDiv [91] has been proposed, which has problems with larger diversity and includes annotations for equations, problem type and grade level. Other large datasets in english language include MathQA [92] and Dolphin18k [93]. Although, these datasets have larger problem set but they are noisy and contain problems of varied difficulty levels.

Chapter 3

Generating Natural Language Attacks in a Hard Label Black Box Setting

3.1 Introduction

In this chapter [61], we study an important and challenging task of attacking natural language processing models in a *hard label black box* setting. We propose a decision-based attack strategy that crafts high quality adversarial examples on text classification and entailment tasks. Our proposed attack strategy leverages population-based optimization algorithm to craft plausible and semantically similar adversarial examples by observing only the top label predicted by the target model. At each iteration, the optimization procedure allow word replacements that maximizes the overall semantic similarity between the original and the adversarial text. Further, our approach does not rely on using substitute models or any kind of training data. We demonstrate the efficacy of our proposed approach through extensive experimentation and ablation studies on *five* state-of-the-art target models across *seven* benchmark datasets. In comparison to attacks proposed in prior literature, we are able to achieve a higher success rate with lower word perturbation percentage that too in a highly restricted setting.

Adversarial attacks are broadly categorized as *white box* and *black box* attacks. White box attacks require access to the target model's architecture, parameters and gradients to craft adversarial examples. Such attacks are expensive to apply and requires access to internal details of the target model which are rarely available in real world applications. Black box attacks are further classified into score-based, transfer-based and decision-based attacks. Score-based attacks generate adversarial examples using the class probabilities or confidence score of the target models. Although score-based attacks do not require detailed model knowledge, the assumption of availability of confidence scores is not realistic. Transfer-based attacks rely on training substitute models with synthetic training data, which is inefficient and computationally expensive. Decision-based attacks generate adversarial examples by observing the top label predicted by the target model and are very realistic.
In this chapter, we focus on the *hard label black box* setting in which the adversary crafts adversarial inputs using only the top label predicted by the target model. Compared to attacks in prior literature, hard-label black box attacks (1) requires no information about the target model's architecture, gradients or even class probability scores, (2) requires no access to training data or substitute models and (3) are

more practical in real-world setting. Due to these constraints, generating adversarial examples under this setting is highly challenging. Besides, none of the attacks proposed in previous works will work in this setting. We tackle this challenging and highly realistic setting by utilizing a population-based optimization procedure that optimizes the objective function by querying the target model and observing the hard-label outputs only. We verify the grammatical correctness and fluency of generated examples through automatic and human evaluation. Our main contributions are as follows:

1. We propose a novel decision-based attack setting and generate plausible and semantically similar adversarial examples for text classification and entailment tasks.

2. Our mechanism successfully generates adversarial examples in a hard-label setting without relying on any sort of training data knowledge or substitute models.

3. Our proposed attack makes use of population-based optimization procedure which maximizes the overall semantic similarity between the original and the adversarial text.

4. In comparison to previous attack strategies, our attack achieves a higher success rate and lower perturbation rate that too in a highly restricted setting.

The *hard label black box* setting has been explored recently in computer vision [53, 54] but to the best of our knowledge we are first to explore it for NLP domain.

3.2 Problem Formulation

Let $\mathbf{F} : \mathcal{X} \rightarrow \mathcal{Y}$, be a target model that classifies an input text sequence $\mathcal{X}$ to a set of class labels $\mathcal{Y}$. Our aim is to craft an adversarial text sequence $\mathcal{X}^*$ that is misclassified by the target model i.e. $\mathbf{F}(\mathcal{X}) \neq \mathbf{F}(\mathcal{X}^*)$ and is semantically similar to the original input $\mathcal{X}$. We obtain $\mathcal{X}^*$ by solving the following constrained-optimization problem:

$$\max_{\mathcal{X}^*} \mathcal{S}(\mathcal{X}, \mathcal{X}^*) \quad s.t. \quad \mathcal{C}(\mathbf{F}(\mathcal{X}^*)) = 1 \tag{3.1}$$

where the function $\mathcal{S}$ computes the semantic similarity between $\mathcal{X}$ and $\mathcal{X}^*$. $\mathcal{C}$ is an adversarial criteria that equals to 1 if $\mathcal{X}^*$ is out of the target model's decision boundary and 0 otherwise. The above equation can be reformulated as:

$$\max_{\mathcal{X}^*} \mathcal{S}(\mathcal{X}, \mathcal{X}^*) \; + \; \delta(\mathcal{C}(\mathbf{F}(\mathcal{X}^*)) = 1) \tag{3.2}$$

where $\delta(x) = 0$ if x is true, otherwise $\delta(x) = -\infty$. We can obtain an adversarial sample $\mathcal{X}^*$ with minimal perturbation by optimizing the objective function in the equation 2. Note that $\mathcal{C}$ is a discontinuous function as the model outputs hard-labels only. This also makes the objective function in equation 2 discontinuous and difficult to optimize.

3.3 Proposed Attack

In a hard label black-box setting, the attacker has no access to model's gradients, parameters or the confidence scores of the target model. Further, the attacker does not have access to the training data on which the target models are trained. To generate a successful attack, we formulate this setting as a constrained optimization problem as shown in equation 3.2. The equation 3.2 optimizes the semantic similarity by querying and observing the final decisions of the target model. Moreover, the outputs of the target model are insensitive to small perturbations as the model returns hard labels only thus posing an ever bigger challenge. We propose a three step strategy to solve this problem. (A) Initialisation — Initialize $\mathcal{X}^*$ outside the target model's decision boundary, (B) Search Space Reduction — Moves $\mathcal{X}^*$ close to the decision boundary and (C) Population-based optimization — Maximizes semantic similarity between $\mathcal{X}$ and $\mathcal{X}^*$ until $\mathcal{X}^*$ is on the target model's decision boundary (Figure **??**).

3.3.1 Initialisation

In order to generate an adversarial example $\mathcal{X}^*$, which is semantically similar to original input $\mathcal{X}$, we restrict the replacement of each word with its top 50 synonyms from the counter-fitted embedding space [50]. Synonyms with part-of-speech (POS) tag different from the original word are filtered out. This ensures that the synonym fits within the context and the sentence is grammatically correct. To optimize the objective function in equation 3.2, $\mathcal{X}^*$ must satisfy the adversarial criteria $\mathcal{C}$. Therefore, $\mathcal{X}^*$ is initialised with a sample that is already out of the target model's decision boundary. This is done by substituting a word in $\mathcal{X}$ with a synonym, sampled randomly from its synonym set $Syn(x_i)$. The above step is repeated for other words in $\mathcal{X}$ until $\mathcal{X}$ moves out of target's model decision boundary or 30% of the words in $\mathcal{X}$ has been substituted (Algorithm 1, lines 3-7). Note that we do not replace a word by a random word or Out of Vocabulary (OOV) token as such a replacement can highly alter the semantics of the text.

3.3.2 Search Space Reduction

Though population-based optimization algorithms are powerful combinatorial optimization techniques, they still slow down and converge to local optima if the size of the search space is large. As shown in Figure 2, substituting more synonyms in $\mathcal{X}^*$ increases the search space exponentially. Therefore, in this step we reduce the substitution count in $\mathcal{X}^*$ by replacing some of the synonyms back with their respective original words. Following steps are used to reduce the substitution count in $\mathcal{X}^*$:

1. Given the initial sample $\mathcal{X}^* = \{x_1, x_2..w_i..x_n\}$ where w_i denotes the synonym of x_i substituted during initialisation. Each synonym w_i is replaced back with its original counterpart x_i (Algorithm 1, line 8-10).

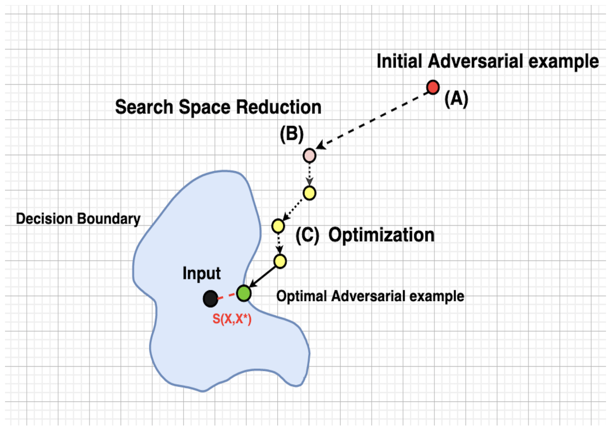


Figure 3.1: Overview of proposed strategy. (A) Adversarial example obtained after initialisation (B) Adversarial sample after search space reduction (C) Optimization steps

2. The text samples which do not satisfy the adversarial criterion are filtered out. From the remaining text samples, each replacement (w_i with x_i) is scored based on the semantic similarity between $\mathcal{X}_i$ and $\mathcal{X}$. All the replacements are sorted in descending order based on this score (Algorithm 1, line 11-13)

3. Synonyms in $\mathcal{X}^*$ are replaced back with their original counterpart in the order decided in step 2 until $\mathcal{X}^*$ satisfies the adversarial criteria (Algorithm 1, line 14-17).

This can be viewed as moving the initial sample $\mathcal{X}^*$ close to the decision boundary of the target model. This process is highly effective as it not only speeds up the optimization algorithm but also prevents it from converging to local optima.

3.3.3 Population Based Optimization

In this section we provide a brief overview of the Genetic Algorithm (GA) and explain the working of our proposed optimization procedure in detail.

3.3.4 Overview

Genetic Algorithm (GA) is a search based optimization technique that is inspired by the process of natural selection — the process that drives biological evolution. GA starts with an initial population of candidate solutions and iteratively evolves them towards better solutions. At each iteration, GA uses a fitness function to evaluate the quality of each candidate. High quality candidates are likely to be selected for generating the next set of candidates through the process of *crossover* and *mutation*. GA has the following four steps:

1. **Initialisation:** GA starts with an initial set of candidates.

Algorithm 1 Initialisation and Search Space Reduction

Input: Test sample $\mathcal{X}$, n word count in $\mathcal{X}$

Output: Adversarial sample $\mathcal{X}^*$

```
1:  $indices \leftarrow Randomly\ select\ 30\%\ positions$
2:  $\mathcal{X}^* \leftarrow \mathcal{X}$
3:  **for** $i$ **in** $indices$ **do**
4:      $w \leftarrow random(Syn(x_i))$    $//\ Sample\ a\ synonym$
5:      $\mathcal{X}^* \leftarrow Replace\ x_i\ with\ w\ in\ \mathcal{X}^*$
6:      **if** $\mathcal{C}(\mathbf{F}(\mathcal{X}^*)) = \mathbf{1}$ **then**
7:          **break**
8:  **for** $i$ **in** $indices$ **do**
9:      $\mathcal{X}_i \leftarrow Replace\ w_i\ with\ x_i\ in\ \mathcal{X}^*$
10:     $scr_i \leftarrow Sim(\mathcal{X}, \mathcal{X}_i)$
11:     **if** $\mathcal{C}(\mathbf{F}(\mathcal{X}_\mathbf{i})) = \mathbf{1}$ **then**
12:         $Scores.insert(scr_i, x_i)$
13: $Sort\ Scores\ by\ scr_i$
14: **for** $x_i$ **in** $Scores$ **do**
15:     $\mathcal{X}_t \leftarrow Replace\ w_i\ with\ x_i\ in\ \mathcal{X}^*$
16:     **if** $\mathcal{C}(\mathbf{F}(\mathcal{X}_\mathbf{t})) = \mathbf{0}$ **then**
17:         **break**
18:     $\mathcal{X}^* \leftarrow \mathcal{X}_t$
19: **return** $\mathcal{X}^*$ $//\ After\ search\ space\ reduction$
```

2. **Selection:** Each candidate is evaluated using a fitness-function. Two candidates (*parents*) are selected based upon their fitness values.

3. **Crossover:** The selected *parents* undergoes crossover to produce the next set of candidates.

4. **Mutation:** The new candidates are mutated to ensures diversity and better exploration of search space. Steps 2-4 are repeated for a specific number of iterations.

We choose GA as an optimization procedure because it's directly applicable to discrete input space. Besides, GA is more intuitive and easy to apply in comparison to other population-based optimization methods. The method proposed in [40] uses the probability scores of the target label for optimizing GA in each iteration of the optimization step. *However, in a hard label black box setting such an* *optimization strategy will fail due to unavailability of probability scores. In this work, GA is used with a completely different motive — we maximize the semantic similarity between two text sequences.* This improves the overall attack success rate and lowers the perturbation percentage that too in a hard label setting where none of the attacks proposed in previous works will work.

3.3.5 Optimization Procedure

Given that the adversarial criteria $\mathcal{C}$ is satisfied for $\mathcal{X}^*$, we now maximize the semantic similarity between $\mathcal{X}$ and $\mathcal{X}^*$ by optimizing equation 3.2. This optimization is achieved by replacing each substituted synonym in $\mathcal{X}^*$ back with the original word or by a better synonym (of the original word) that results in higher overall semantic similarity. We now define three operations *mutation*, *selection* and *crossover* that constitutes one iteration of the optimization step.

3.3.5.1 Mutation

Given input $\mathcal{X}^* = \{x_1, w_2, w_3, x_4, x_5...x_n\}$ and $idx \in [0, n]$, the position where mutation needs to be applied, x_i is the original word and w_{idx} is the synonym substituted in Algorithm 1. This step aims to find a better replacement for the substituted synonym w_{idx} in $\mathcal{X}^*$ such that (1) semantic similarity between $\mathcal{X}$ and $\mathcal{X}^*$ improves and (2) $\mathcal{X}^*$ satisfies the adversarial criteria. To achieve this, first the substituted synonym w_{idx} in $\mathcal{X}^*$ is replaced back with the original word x_{idx}. If the new text sequence satisfies the adversarial criteria, than x_{idx} is selected as the final replacement for w_{idx}. Otherwise, the substituted synonym w_{idx} is replaced with each synonym from the synonym set $Syn(x_{idx})$. This results in a set $\mathcal{T} = \{\mathcal{X}_1^*, \mathcal{X}_2^*...\mathcal{X}_l^*\}$ where l in the number of synonyms of x_{idx} and $\mathcal{X}_j^* = \{x_1, w_2, s_{idx}, x_4, x_5...x_n\}$ where $j \in [1, l]$ is the text sample obtained after replacing w_{idx} with a synonym s_{idx} from the set $Syn(x_{idx})$. The generated samples which do not satisfy the adversarial criteria are filtered out from $\mathcal{T}$. From the remaining samples, all the samples which improves the overall semantic similarity score (equation 3.3) with the original input $\mathcal{X}$ is selected.

$$Sim(\mathcal{X}, \mathcal{X}_j^*) >= Sim(\mathcal{X}, \mathcal{X}^*) \quad for\; \mathcal{X}_j^* \in \mathcal{T} \tag{3.3}$$

If there are multiple samples in $\mathcal{T}$ which improves the semantic similarity than the sample with the highest semantic similarity score is selected as the final mutated sample.

$$candidate = \underset{\mathcal{X}_j^* \in \mathcal{T}}{\arg\max}\; Sim(\mathcal{X}, \mathcal{X}_j^*) \tag{3.4}$$

Note that for point 6 in optimization steps, the input to mutation will be a candidate from population set $\mathcal{P}^i$.

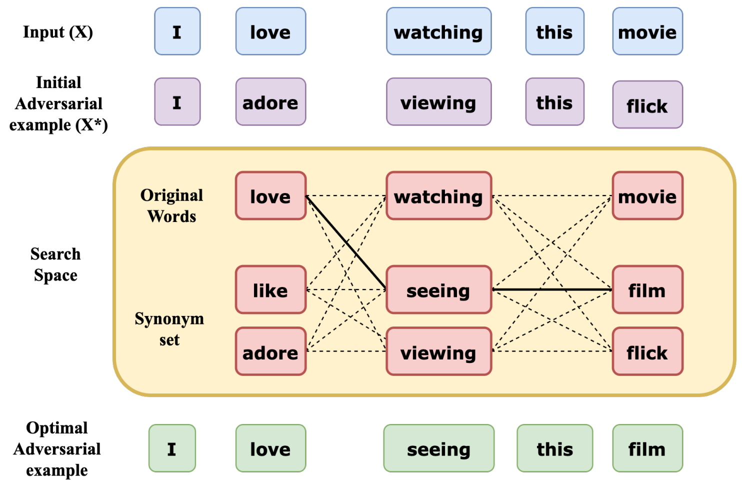


Figure 3.2: Search space of an adversarial sample $\mathcal{X}^*$. Dotted lines shows all possible combinations. Bold lines shows the selected combination which has the highest semantic similarity with $\mathcal{X}$ and satisfies the adversarial criteria $\mathcal{C}$.

3.3.5.2 Selection

Given a population set $\mathcal{P}^i = \{c_0^i, c_1^i, c_2^i ... c_{\mathcal{K}}^i\}$ where $\mathcal{K}$ represents the population size. This step samples two candidates c_p^i, c_q^i where $p, q \in [0, \mathcal{K}]$ based upon the value assigned by the fitness function. As we optimize the semantic similarity of an adversarial sequence with the original input we take the semantic similarity between the adversarial and the original text as our fitness function.

$$z_y = Sim(\mathcal{X}, c_y^i) \quad where\ c_y^i \in\ \mathcal{P}^i; y \in [0, \mathcal{K}] \tag{3.5}$$

This will allow candidates with higher semantic similarity scores to be selected as *parents* for the crossover step. c_p^i, c_q^i are sampled from $\mathcal{P}^i$ with probability proportional to $\phi(z)$.

$$\phi(z) = \frac{exp(z)}{\sum_{y=0}^{\mathcal{K}} exp(z_y)} \tag{3.6}$$

3.3.5.3 Crossover

Given c_p^i, c_q^i this step generates a new $candidate$ text sequence which satisfies the adversarial criteria. It randomly selects a word for each position of $candidate$ from c_p^i and c_q^i. Crossover is repeated multiple times to ensure exploration of various combinations in the search space.

$$c_p^i = \{u_0^1, u_1^1 ... u_n^1\} \tag{3.7}$$

$$c_q^i = \{u_0^2, u_1^2 ... u_n^2\} \tag{3.8}$$

$$candidate = \{rand(u_0^1, u_0^2) ... rand(u_n^1, u_n^2)\} \tag{3.9}$$

where u_0^1, u_0^2 represents the first word in c_p^i and c_q^i respectively and $rand(u_0^1, u_0^2)$ randomly selects a word.

3.3.5.4 Optimization Steps

For an adversarial text $\mathcal{X}^*$ generated from Algorithm 1, GA based optimization executes the following steps.

1. Initally, all the indices of the substituted synonyms in $\mathcal{X}^*$ is maintained in a set pos.

2. For an index idx in pos, $\mathcal{X}^*$ is mutated to generate an adversarial sample $candidate$ (equation 4). When executed for all idx in pos, we get a $candidate$ corresponding to each idx which constitutes an initial population $\mathcal{P}^0$ as shown in equations 3.10 and 3.11. $\mathcal{K}$ is population size.

$$c_m^0 = Mutation(\mathcal{X}^*, idx); idx \in pos;\ m \in [0, \mathcal{K}] \tag{3.10}$$

$$\mathcal{P}^0 = \{c_0^0, c_1^0, c_2^0 .... c_{\mathcal{K}}^0\} \tag{3.11}$$

3. A candidate $\mathcal{X}_{final}$ with the highest semantic similarity with $\mathcal{X}$ is selected from population $\mathcal{P}^i$.

$$\mathcal{X}_{final} = \underset{c_m^i\, \in\, \mathcal{P}^i}{\arg\max}\ Sim(\mathcal{X}, c_m^i) \tag{3.12}$$

4. A candidate pair (parents) is sampled independently from $\mathcal{P}^i$ with probability proportional to $\phi(z)$.

$$c_p^i, c_q^i = Selection(\mathcal{P}^i) \tag{3.13}$$

5. The two candidates c_p^i, c_q^i undergoes crossover $\mathcal{K} - 1$ times to generate the next set of candidates.

$$c_m^{i+1} = Crossover(c_p^i, c_q^i) \tag{3.14}$$

   where c_m^{i+1} represents the mth candidate in $i + 1$ th step. Candidates which does not satisfy the adversarial criteria $\mathcal{C}$ and also have less semantic similarity score than $\mathcal{X}_{final}$ are filtered out.

6. For each candidate c_m^{i+1} obtained from step 5, an index idx is randomly sampled from pos. If the word at index idx in c_m^{i+1} has not yet been replaced back by the original word than c_m^{i+1} is mutated at index idx. Otherwise c_m^{i+1} is passed as such to the next population set $\mathcal{P}^{i+1}$.

$$c_m^{i+1} = Mutation(c_m^{i+1}, idx);\ m \in [0, \mathcal{K}] \tag{3.15}$$

$$\mathcal{P}^{i+1} = \{\mathcal{X}_{final}, c_0^{i+1}, c_1^{i+1}, c_2^{i+1} ... c_{\mathcal{K}}^{i+1}\} \tag{3.16}$$

The steps 3 to 6 are than repeated for the next population set $\mathcal{P}^{i+1}$. The maximum number of iterations T for the steps 3-6 are set to 100. Further, each index of the substituted synonym is allowed to be mutated at most $\lambda = 25$ times as it avoids the GA to converge to local optimum.

3.4 Experiments

We perform experiments across *seven* benchmark datasets on *five* target models and compare our proposed attack strategy with *seven* state-of-the-art baselines.

3.4.1 Datasets

(1) *AG News* is a multiclass news classification dataset. The description and title of each article is concatenated following [94]. (2) *Yahoo Answers* is a document level topic classification dataset. The question and top answer are concatenated following [94]. (3) *MR* is a sentence level binary classification of movie reviews [95]. (4) *IMDB* is a document level binary classification dataset of movie reviews [96]. (5) *Yelp Reviews* is a sentiment classification dataset [94]. Reviews with rating 1 and 2 are labeled negative and 4 and 5 positive as in [43]. (6) *SNLI* is a dataset consisting of hypothesis and premise sentence pairs. [97]. (7) *MultiNLI* is a multi-genre NLI corpus [98].

Dataset	**Train**	**Test**	**Classes**	**Avg. Len**
AG News	120K	7.6K	4	43
Yahoo	12K	4K	10	150
MR	9K	1K	2	20
IMDB	12K	12K	2	215
Yelp	560K	18K	2	152
SNLI	120K	7.6K	3	8
MultiNLI	12K	4K	3	10

Table 3.1: Statistics of all datasets

3.4.2 Target Models

We attacked WordCNN [14], WordLSTM [10] and BERT base-uncased [19] for text classification. For WordLSTM, a single layer bi-direction LSTM with 150 hidden units was used. In WordCNN windows of sizes 3, 4 and 5 each having 150 filters was used. For both WordCNN and WordLSTM a dropout rate of 0.3 and 200 dimensional Glove word embedding were used. For textual entailment task, we attacked ESIM [99], InferSent [100] and BERT base-uncased. The original accuracies of all the models are shown in Table 2.

3.4.3 Baselines

(1) *PSO* is a score-based attack that uses sememe-based substitution and particle swarm optimization [44]. (2) *TextFooler* uses target model confidence scores to rank words and replaces those with synonyms [43]. (3) *PWWS* ranks word based on model confidence scores and finds substitutes using

Dataset	Attack	BERT				WordLSTM				WordCNN			
		Orig.%	**Acc.%**	**Pert.%**	**I%**	**Orig.%**	**Acc.%**	**Pert.%**	**I%**	**Orig.%**	**Acc.%**	**Pert.%**	**I%**
MR	TF	86.00	11.5	16.7	1.26	80.7	3.1	14.90	1.04	78.00	2.8	14.3	**1.27**
	Ours		**7.4**	**10.7**	**1.04**		**2.8**	**12.2**	**0.93**		**2.5**	**11.9**	1.30
IMDB	TF	90.00	13.6	6.10	0.5	89.8	0.3	5.1	0.53	89.20	**0.0**	3.50	0.40
	Ours		**1.1**	**3.13**	**0.36**		**0.2**	**2.9**	**0.27**		**0.0**	**2.8**	**0.37**
Yelp	TF	96.50	6.6	13.9	1.01	95.00	2.1	10.76	1.07	93.80	**1.1**	8.30	0.84
	Ours		**5.2**	**6.37**	**0.62**		**3.2**	**6.7**	**0.62**		**1.1**	**6.44**	**0.78**
AG	TF	94.20	12.5	22.0	1.58	91.30	**3.8**	18.6	1.35	91.5	1.5	15.0	0.91
	Ours		**5.8**	**12.2**	**0.74**		4.1	**12.9**	**0.83**		**1.0**	**10.2**	**0.90**
Yahoo	TF	79.10	18.2	17.7	1.72	73.7	16.6	18.41	0.94	71.1	9.2	15.0	0.80
	Ours		**8.0**	**4.5**	**0.44**		**4.2**	**6.3**	**0.67**		**2.4**	**6.1**	**0.65**

Dataset	Attack	BERT				InferSent				ESIM			
		Orig.%	**Acc.%**	**Pert.%**	**I%**	**Orig.%**	**Acc.%**	**Pert.%**	**I%**	**Orig.%**	**Acc.%**	**Pert.%**	**I%**
SNLI	TF	89.1	**4.0**	18.5	9.7	84.0	3.5	18.0	7.7	86.0	5.1	18.1	8.6
	Ours		5.2	**16.7**	**3.70**		**4.1**	**18.2**	**3.70**		**4.1**	**17.2**	**3.62**
MNLI	TF	85.1	9.6	15.4	7.7	70.9	6.7	14.0	4.7	77.9	7.7	14.5	**1.6**
	Ours		**5.2**	**13.5**	**2.79**		**5.1**	**13.2**	**2.98**		**5.9**	**13.1**	2.76
MNLIm	TF	82.1	8.3	14.6	7.3	69.6	6.9	14.6	3.6	75.8	7.3	14.6	2.6
	Ours		**4.1**	**12.71**	**2.56**		**4.5**	**12.8**	**3.1**		**4.0**	**12.6**	**2.4**

Table 3.2: Comparison with TextFooler (TF). Orig.% is the original accuracy, Acc.% is the after attack accuracy, Pert.% is the average perturbation rate and I% is the average grammatical error increase rate. Mnlim is the mis-matched version of MNLI.

WordNet [42]. (4) *TextBugger* finds important sentences using the confidence scores of the target model and replaces words in those sentences with synonyms [41]. (5) *Genetic Attack* is a score-based attack which crafts attacks using a population-based optimization algorithm [40]. (6) *GANs* is a decision-based attacking strategy that generates adversarial examples using GANs on textual entailment task [47]. (7) *DeepRL* relies on substitute models and reinforcement learning to generate attacks [48]

3.4.3.1 Evaluation Metrics

We use *after attack accuracy* to evaluate the performance of our proposed attack. It refers to the accuracy of the target model obtained on the generated adversarial examples. High difference between the original and after attack accuracy represents a more successful attack. We use *perturbation rate* and *grammatical correctness* to evaluate the quality of generated adversarial examples. *Perturbation rate* refers to the number of words substituted in the input to generate an adversarial example. A higher perturbation rate degrades the quality of generated adversarial example. For *grammatical correctness* we record the grammatical error rate increase between the original and the final adversarial example.

Dataset	Model	Attack	Succ.%	Pert.%
IMDB	**WordLSTM**	TextBugger	86.7	6.9
		Genetic	97.0	14.7
		PSO	**100.0**	3.71
		Ours	99.8	**2.9**
	WordCNN	PWWS	95.5	3.8
		DeepRL	79.4	-
		Ours	**100.0**	**2.8**
	BERT	PSO	98.7	3.69
		Ours	**98.9**	**3.13**
SNLI	**Infersent**	PSO	73.4	**11.7**
		Genetic	70.0	23.0
		GANs	69.6	-
		Ours	**96.6**	17.7
	BERT	PSO	78.9	**11.7**
		Ours	**94.8**	16.7
AG	**WordCNN**	PWWS	43.3	16.7
		Ours	**99.0**	**10.2**
Yahoo	**WordCNN**	PWWS	42.3	25.4
		Ours	**97.6**	**6.1**

Table 3.3: Comparison with other baselines. Succ.% is attack success rate and Pert.% is average word perturbation rate.

Examples	**Prediction**
Highly [Exceedingly] watchable stuff.	**Positive $\rightarrow$ Negative**
It's weird, wonderful, and not **necessarily [definitely]** for kids.	**Negative $\rightarrow$ Positive**.
Could i use both Avast and Avg to protect my **computer [machinery]**? I recommend the free version of Avg antivirus for home users.	**Technology $\rightarrow$ Music.**
Premise: Larger ski resorts are 90 minutes away. **Hypothesis**: If we travel for 90 minutes, we could arrive at larger **ski [skating]** resorts.	**Entailment $\rightarrow$ Neutral.**
Premise: A portion of the nation's income is saved by allowing for capital investment. **Hypothesis**: The nation's income is divided into portions **[fractions]**.	**Entailment $\rightarrow$ Neutral.**

Table 3.4: Adversarial samples generated on BERT. The actual word is bold and substituted word are bold and in square brackets.

We used Language-Tool[1] to calculate the grammatical error rate of each generated adversarial text. The adversarial examples with very high perturbation rate ($> 25\%$) are filtered out. For all the evaluation metrics, we report the average score across all the generated adversarial examples on each dataset. We also conducted human evaluation to verify the quality of adversarial examples.

[1]`https://www.languagetool.org/`

3.4.3.2 Experimental Settings

We use Universal Sequence Encoder (USE) [101] to compute the semantic similarity between the original and adversarial example. We filter out stop words using NLTK and use Spacy for POS tagging. The target models are attacked on a set of 1000 examples, sampled from the test set of each dataset. To ensure fair comparison these are the same set of examples used in [40, 43]. The parameters of GA, $\mathcal{K}$ and λ were set to 30 and 25 respectively. The maximum iterations T is set to 100. From each dataset, we held-out 10% data for validation set, for tuning the hyper-parameters.

3.4.3.3 Attack Performance

Table 3.2 and 3.3 shows that our attack achieves more than 90% success rate on classification and entailment tasks. In comparison to TextFooler, our attack reduces both the perturbation rate and after attack accuracy by atleast 33% and 32% respectively across all datasets and target models. Further, it reduces the grammatical error rate by 27%. When compared to PSO, on IMDB and SNLI, our attack achieves 10% more success rate with lesser perturbation rate that too in a highly restricted setting. Table 3 shows that our attack outperforms other baselines in terms of success rate and perturbation rate. Table 3.4 shows the adversarial examples generated effectively on BERT.

Ablation Study	Metric	IMDB	Yelp	SNLI
no SSR and GA	Pert%	20.1	24.3	34.6
	Sim	0.6	0.57	0.22
only GA	Pert%	4.2	7.2	18.0
	Sim	0.81	0.74	0.37
only SSR	Pert%	6.0	9.7	21.0
	Sim	0.80	0.74	0.26
both SSR and GA	Pert%	**3.1**	**6.7**	**16.7**
	Sim	**0.89**	**0.80**	**0.45**

Table 3.5: Importance of the search space reduction (SSR) and GA step. Pert.% is the average perturbation rate and Sim is the average semantic similarity.

3.5 Ablation Study

Importance of Search Space Reduction: To study the significance of search space reduction we executed GA directly after the initialisation step. Table 3.5 shows the results obtained on BERT. On an average the perturbation rate increased by 1.2% and the semantic similarity dropped by 0.1. This is

Dataset	Pert.%		Sim	
	with ran	w/o ran	with ran	w/o ran
IMDB	3.7	**3.1**	0.81	**0.89**
Yelp	15.3	**6.7**	0.72	**0.80**
MR	18.0	**10.7**	0.53	**0.67**

Table 3.6: Effect of random initialisation (ran). Pert.% is average perturbation and Sim is the average semantic similarity

because GA based optimization converges to a local optimum in most of the cases when search space is large. Further on an average, GA optimization procedure slows down by atleast five times due to large search space.

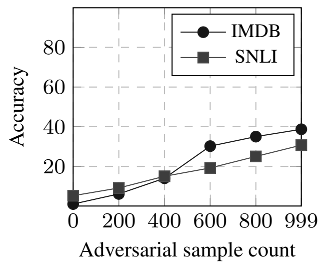
Importance of Genetic Algorithm: To study the importance of GA, we compare perturbation rate and semantic similarity both with and without GA based optimization. Table 3.5 demonstrates the results obtained on BERT. By using GA, the perturbation rate is reduced by 4.4% and the semantic similarity improves by 0.16. Table 5 also shows the perturbation rate and semantic similarity after initialisation step. On an average, the perturbation is 20% higher and and semantic similarity is 0.25 lower. This highlights the combined effect of both GA and search space reduction step to find optimal adversarial examples. We obtained similar results across all datasets and target models.

3.6 Analysis

Importance of Synonym based Initialisation: During initialisation, we replace each word in $\mathcal{X}$ randomly with it's synonym, sampled from top 50 synonyms in the counter-fitted space. To verify its effectiveness, we remove this constraint and allow the word to be replaced with a random word from the vocabulary. Results achieved on BERT are shown in Table 3.6. On an average, the semantic similarity decreases by 0.1 and the perturbation rate increases by 2.4%.

Transferability: An adversarial example is called *transferable* if it's generated against a particular target model but successfully attacks other target models as well. We evaluate the transferability of adversarial attacks generated on IMDB and SNLI datasets. The results are shown in Table 3.7. A lower accuracy of a target model demonstrates high transferability. Adversarial examples generated by our attacks show better transferability when compared to prior attacks.

Adversarial Training: We generated adversarial examples using the 10% samples from the training set of IMDB and SNLI datasets. We augmented the generated adversarial examples with the original training set of the respective datasets and re-trained BERT. We than again attacked BERT with our attack strategy. The results are shown in Figure 3.3. The after attack accuracy and perturbation rate increases by 15% and 10% respectively. This shows that by augmenting adversarial samples to the training data,



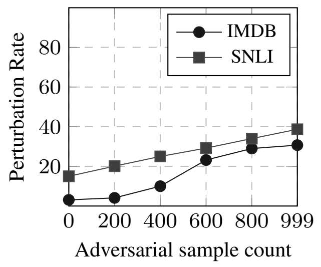


Figure 3.3: Demonstrates increase in after attack accuracy and perturbation as more adversarial samples are augmented.

the target models becomes more robust to attacks.

Human Evaluation: To validate and access the quality of adversarial samples, we randomly sampled 25% of the adversarial examples from IMDB and SNLI datasets. The true class labels of these samples were kept hidden and the evaluators were asked to classify them. We found 96% adversarial examples in IMDB and 92% in SNLI having the same classification label as that of their original samples. The evaluators were asked to score each adversarial example on grammatical correctness and semantic similarity with the original example. They were asked to score each example from 1 to 5 based on grammatical correctness and assign a score of 0, 0.5 and 1 for semantic similarity. Table 3.8 shows the evaluation results of attacks generated against BERT.

Model	**BERT**	**W-CNN**	**W-LSTM**
BERT	-	85.0	86.9
W-CNN	84.6	-	79.1
W-LSTM	80.8	73.6	-

Model	**BERT**	**ESIM**	**Infersent**
BERT	-	53.0	38.5
ESIM	54.9	-	38.5
Infersent	67.4	69.5	-

Table 3.7: Transferability on IMDB (upper half) and SNLI (lower half) datasets. Row i is the model used to generate attacks and column j is the model which was attacked.

Evaluation criteria	**IMDB**	**SNLI**
Grammatical Correctness	4.44	4.130
Semantic Similarity	0.93	0.896

Table 3.8: Demonstrates scores given by evaluators

Chapter 4

A Strong Baseline for Query Efficient Attacks in a Black Box Setting

4.1 Introduction

Existing black box search methods have achieved high success rate in generating adversarial attacks against NLP models. However, such search methods are inefficient as they do not consider the amount of queries required to generate adversarial attacks. Also, prior attacks do not maintain a consistent search space while comparing different search methods. In this chapter, we propose a query efficient attack strategy to generate plausible adversarial examples on text classification and entailment tasks. Our attack jointly leverages attention mechanism and locality sensitive hashing (LSH) to reduce the query count. We demonstrate the efficacy of our approach by comparing our attack with *four* baselines across *three* different search spaces. Further, we benchmark our results across the same search space used in prior attacks. In comparison to attacks proposed, on an average, we are able to reduce the query count by 75% across all datasets and target models. We also demonstrate that our attack achieves a higher success rate when compared to prior attacks in a limited query setting.

Almost all the prior black box attacks consists of two major components (1) *search space* and (2) *search method*. A *search space* is collectively defined by a set of transformations (usually synonyms) for each input word and a set of constraints (e.g., minimum semantic similarity, part-of-speech (POS) consistency). The synonym set for each input word is generated either from the nearest neighbor in the counter-fitted embedding space or from a lexical database such as HowNet [102] or WordNet [49]. The search space is variable and can be altered by either changing the source used to generate synonyms or by relaxing any of the above defined constraints.

A *search method* is a searching algorithm used to find adversarial examples in the above defined search space. Given an input with $\mathcal{W}$ words and each word having $\mathcal{T}$ possible substitutes, the total number of perturbed text inputs is $(\mathcal{T}+1)^{\mathcal{W}}$. Given this exponential size, the search algorithm must be efficient and exhaustive enough to find optimal adversarial examples from the whole search space.

Black box attacks proposed in [40, 44] employ combinatorial optimization procedure as a search method to find adversarial examples in the above defined search space. Such methods are extremely slow and require massive amount of queries to generate adversarial examples. Attacks proposed in [42, 43] search for adversarial examples using word importance ranking which first rank the input words and than substitutes them with similar words. The word importance ranking scores each word by observing the change in the confidence score of the target model after that word is removed from the input (or replaced with UNK token). Although, compared to the optimization based methods, the word importance ranking methods are faster, but it has some major drawbacks – (1) each word is ranked by removing it from the input (or replacing it with a UNK token) which therefore alter the semantics of the input during ranking, (2) it not clear whether the change in the confidence score of the target model is caused by the removal of the word or the modified input and (3) this ranking mechanism is inefficient on input of larger lengths.

In general, their exists a trade off between the attack success rate and the number of queries. A high query search method generates adversarial examples with high success rate and vice-versa. All such prior attacks are not efficient and do not take into consideration the number of queries made to the target model to generate an attack. Such attacks will fail in real world applications where there is a constraint on the number of queries that can be made to the target model.

To compare a new search method with previous methods, the new search method must be benchmarked on the same search space used in the previous search methods. However, a study conducted in [51] have shown that prior attacks often modify the search space while evaluating their search method. This does not ensure a fair comparison between the search methods because it is hard to distinguish whether the increase in attack success rate is due to the improved search method or modified search space. For example, [52] compares their search method with [40] where the former uses Universal Sentence Encoder (USE) [101] and the latter use language model as a constraint. Also, all the past works evaluate their search methods only on a single search space. In this chapter [62], we address the above discussed drawbacks through following contributions:

1. We introduce a *novel ranking mechanism* that takes significantly less number of queries by jointly leveraging word attention scores and LSH to rank the input words; without altering the semantics of the input.

2. We call for unifying the evaluation setting by benchmarking our search method on the same search space used in the respective baselines. Further, we evaluate the effectiveness of our method by comparing it with *four* baselines across *three* different search spaces.

3. On an average, our method is 50% faster as it takes 75% lesser queries than the prior attacks while compromising the attack success rate by less than 2%. Further, we demonstrate that our search method has a much higher success rate than compared to baselines in a limited query setting.

4.2 Proposed Approach

Given a target model $\mathbf{F} : \mathcal{X} \rightarrow \mathcal{Y}$, that maps the input text sequence $\mathcal{X}$ to a set of class labels $\mathcal{Y}$. Our goal is to generate an adversarial text sequence $\mathcal{X}_{\mathcal{ADV}}$ that belongs to any class in $\mathcal{Y}$ except the original

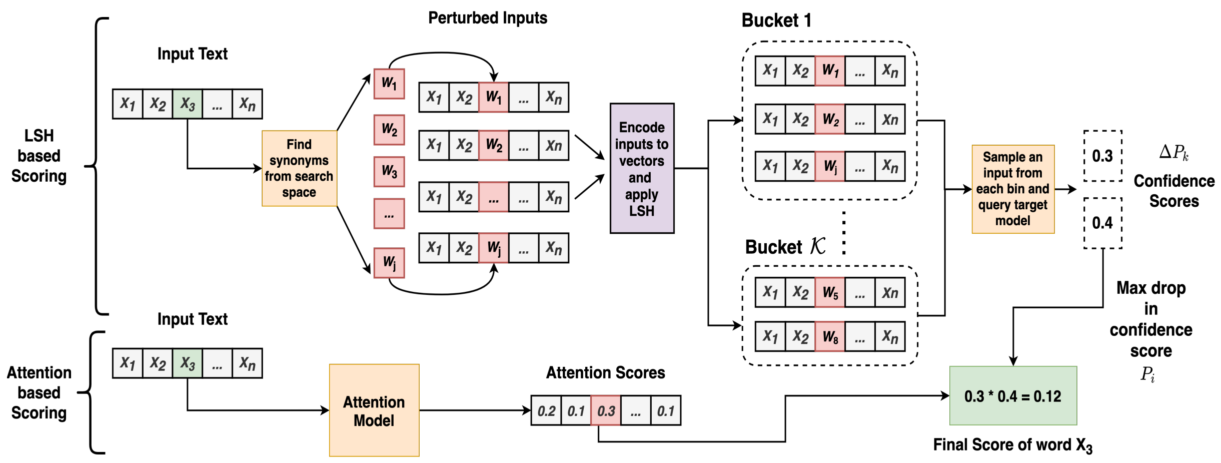


Figure 4.1: Scoring of each input word using attention mechanism and Locality Sensitive Hashing.

class of $\mathcal{X}$ i.e. $\mathbf{F}(\mathcal{X}) \neq \mathbf{F}(\mathcal{X}_{\mathcal{ADV}})$. The input $\mathcal{X}_{\mathcal{ADV}}$ must be generated by substituting the input words with their respective synonyms from a chosen search space.
Our search method consists of two steps (1) Word Ranking – ranks all words in the input text and (2) Word Substitution – substitutes input words with their synonyms in the order-of-rank (step 1).

4.2.1 Word Ranking

Recent studies [103] have shown evidence that certain words in the input and their replacement can highly influence the final prediction of DNNs. Therefore, we score each word based upon, (1) *how important it is for the final prediction* and (2) *how much its replacement with a similar word can alter the final prediction* of the target model. We use *attention mechanism* to select important words for classification and employ *LSH* to capture the impact of replacement of each word on the prediction of target model. Figure 4.1 demonstrates the working of the word ranking step.

4.2.1.1 Attention based scoring

Given an input $\mathcal{X}$, this step assigns high score to those influential words which impact the final outcome. The input sequence $\mathcal{X} = \{x_1, x_2..x_n\}$ is passed through a pre-trained attention model F_{attn} to get attention scores α_i for each word x_i. The scores are computed using Hierarchical Attention Networks (HAN) [104] and Decompose Attention Model (DA) [105] for text classification and entailment tasks respectively. Note, instead of querying the target model every time to score a word, this step scores all words together in a single pass (inferring the input sample by passing it through attention model), thus, significantly reducing the query count. Unlike prior methods, we do not rank each word by removing it from the input (or replacing it with a UNK token), preventing us from altering semantics of the input.

4.2.1.2 LSH based Scoring

This step assigns high scores to those words whose replacement with a synonym will highly influence the final outcome of the model. It scores each word based on the change in confidence score of the target model, when it is replaced by its substitute (or synonym) word. But computing the change in confidence score for each synonym for every input word, significantly large number of queries are required. Therefore, we employ LSH to solve this problem. LSH is a technique used for finding nearest neighbours in high dimensional spaces. It takes an input, a vector x and computes its hash $h(x)$ such that similar vectors gets the same hash with high probability and dissimilar ones do not. LSH differs from cryptographic hash methods as it aims to maximize the collisions of similar items. We leverage Random Projection Method (RPM) [106] to compute the hash of each input text.

4.2.1.3 Random Projection Method (RPM)

Let us assume we have a collection of vectors in an m dimensional space R^m. Select a family of hash functions by randomly generating a spherically symmetrical random vector r of unit length from the m dimensional space. Then the hash function $h_r(u)$ is defined as:

$$h_r(u) = \begin{cases} 0 & r.u < 0 \\ 1 & r.u \geq 0 \end{cases} \tag{4.1}$$

Repeat the above process by generating d random unit vectors $\{r_0, r_1..r_d\}$ in R^m. The final hash $\bar{u}$ for each vector u is determined by concatenating the result obtained using on all d vectors.

$$\bar{u} = \{h_{r1}(u), h_{r2}(u)..h_{rd}(u)\}. \tag{4.2}$$

The hash of each vector u is represented by a sequence of bits and two vectors having same hash are mapped to same bucket in the hash table. Such a process is very efficient in finding nearest neighbor in high dimensional spaces as the hash code is generated using only the dot product between two matrices. Also, it is much easier to implement and simple to understand when compared to other nearest neighbour methods. We use the above process to score each word as follows:

1. First, an input word x_i is replaced with every synonym from its synonym set $S(x_i)$ resulting in perturbed sequence $\mathcal{X}_{ij} = \{x_1...w_j...x_n\}$, where i is the substituted index and j is the jth synonym from $S(x_i)$ i.e. $w_j \in S(x_i)$. Perturbed sequences not satisfying the search space constraints (Table 4.2) are filtered.

2. The remaining perturbed inputs are passed through a sentence encoder (USE) which returns a vector representation $\mathcal{V}_j$ for each perturbed input. Then we use LSH as described above to compute the hash of each vector. The perturbed inputs having the same hash are mapped to same bucket of

the hash table.

$$\mathcal{V}_j = \mathbf{encode}(\mathcal{X}_{ij}) \quad \forall j;\ j \in [1, \mathcal{T}] \tag{4.3}$$

$$\mathcal{B} = \mathbf{LSH}(\mathcal{V}_j, \mathcal{X}_{ij}) \quad \forall j;\ j \in [1, \mathcal{T}] \tag{4.4}$$

were $\mathcal{T}$ are the number of perturbed inputs obtained for each word and $\mathcal{B} = \{b_0, b_1 ... b_{\mathcal{K}}\}$ are the buckets obtained after LSH in a hash table, $\mathcal{K}$ being the number of buckets.

3. As each bucket contains similar perturbed inputs, a perturbed input is sampled randomly from each bucket and is passed to the target model **F**. The maximum change in the confidence score of the target model among all these fed inputs is the score for that index.

$$\mathcal{V}_k^*, \mathcal{X}_k^* = sample(b_k)\ \forall k;\ k \in [0, \mathcal{K}] \tag{4.5}$$

$$\Delta P_k = \mathbf{F}(\mathcal{X}) - \mathbf{F}(\mathcal{X}_k^*)\ \forall k;\ k \in [0, \mathcal{K}] \tag{4.6}$$

$$P_i = \max(\Delta P_k) \qquad k \in [0, \mathcal{K}] \tag{4.7}$$

The steps 1 to 3 are repeated for all the indices in $\mathcal{X}$. LSH maps highly similar perturbed inputs to the same bucket, and as all such inputs are similar, they will impact the target model almost equally. Therefore, instead of querying the target model **F** for every input in the bucket, we sample an input randomly from each bucket and observe its impact on the target model **F**. This will reduce the query count from being proportional to number of synonyms of each word to minimum number of buckets obtained after LSH.

4.2.1.4 False Negative error rate of LSH

Although LSH is efficient for finding nearest neighbour, there is still a small probability that similar perturbed inputs get mapped to different buckets. Therefore to reduce this probability, we conduct multiple rounds of hashing, $L = 15$, each round with a different family of hash functions and choose the round which has the most collisions i.e. round having minimum buckets. [106] establishes an upper bound on the false negative error rate of LSH i.e two highly similar vectors are mapped to different buckets. The upper bound on the false negative error rate of LSH given by (for more details refer [106])

$$(1 - P^d)^L \tag{4.8}$$

$$where\ P \ = \ 1 - \frac{\theta(u, v)}{\pi} \tag{4.9}$$

This shows that for given values of L and d, LSH maps similar vectors to the same bucket with very high probability. As LSH maps similar inputs to the same bucket with high probability, it cuts down the synonym search space drastically, thus reducing the number of queries required to attack. The dimension of the hash function d used in equation 4.2 is set to 5 and is same across all rounds.

4.2.1.5 Final Score Calculation

After obtaining the attention scores α_i and the scores from synonym words P_i for each index (calculated using LSH), we multiply the two to get the final score $score_i = \alpha_i * P_i$ for each word. All the words are sorted in descending order based upon the score $score_i$. Algorithm 2 demonstrates all the steps for word ranking.

Algorithm 2 Word Ranking

Input: Test sample $\mathcal{X}$

Output: $\mathcal{W}$ containing score of each word x_i

```
1: $F_{attn} \leftarrow HAN()\ or\ DA()$
2: $\alpha \leftarrow F_{attn}(\mathcal{X})$
3: for $x_i$ in $\mathcal{X}$ do
4:     $S \leftarrow Synonyms(x_i)$
5:     for $w_j$ in $S$ do
6:         $\mathcal{X}_{ij} \leftarrow Replace\ x_i\ with\ w_j\ in\ \mathcal{X}$
7:         $\mathcal{V}_j \leftarrow encode(\mathcal{X}_{ij})$
8:         $\{b_1..b_{\mathcal{K}}\} = \mathbf{LSH}(\mathcal{V}_j, \mathcal{X}_{ij})$
9:     for $k = 1\ to\ \mathcal{K}$ do
10:        $\mathcal{V}_k^*, \mathcal{X}_k^* \leftarrow sample(b_k)$
11:        $\Delta P_k = \mathbf{F}(\mathcal{X}) - \mathbf{F}(\mathcal{X}_k^*)$
12:    $P_i = \max(\Delta P_k)$
13:    $score_i \leftarrow \alpha_i * P_i$
14:    $\mathcal{W}.insert((score_i, x_i))$
15: $Sort\ \mathcal{W}\ by\ score_i\ in\ descending\ order$
```

4.2.2 Word Substitution

We generate the final adversarial example for the input text by perturbing the words in the order retrieved by the word ranking step. For each word x_i in $\mathcal{W}$, we follow the following steps.

1. Each synonym w_j from the synonym set $S(x_i)$ is substituted for x_i which results in a perturbed text $X'_j = \{x_1..w_j..x_n\}$. The perturbed texts which do not satisfy the constraints imposed on the search space (Table 4.2) are filtered (Algorithm 3, lines $1-7$).

2. The remaining perturbed sequence(s) are fed to the target model $\mathbf{F}$ to get the class label y_{new} and its corresponding probability score P_{new}. The perturbed sequence(s) which alters the original class label y_{orig} is chosen as the final adversarial example $\mathcal{X_{ADV}}$. In case the original label does not change, the perturbed sequence which has the minimum P_{new} is chosen (Algorithm 3, lines $7-14$).

The steps $1-2$ in Algorithm 3 are repeated on the chosen perturbed sequence for the next ranked word. Note, we only use LSH in the word ranking step, because in ranking we need to calculate scores for all the words in the input and so we need to iterate over all possible synonyms of every input word. However, in the word substitution step we replace only one word at a time and the substitution step stops when we get an adversarial example. As the number of substitutions are very less (see perturbation rate in Table 4.1) to generate adversarial example, the substitution step iterate over very less words when compared to ranking step.

Algorithm 3 Word Substitution

```
Input: Test sample $\mathcal{X}$, Ranked words $\mathcal{W}$
Output: Adversarial text $\mathcal{X}_{ADV}$
1: $\mathcal{X}_{ADV} \leftarrow \mathcal{X}$
2: $y_{orig}\ ,\ P_{orig} \leftarrow \mathbf{F}(\mathcal{X_{ADV}})$
3: $P_{best} \leftarrow P_{orig}$
4: for $(score_i, x_i)\ in\ \mathcal{W}$ do
5:     $S \leftarrow\ Synonyms(x_i)$
6:     for $w_j\ in\ S$ do
7:         $X'_j \leftarrow\ Replace\ x_i\ with\ w_j$
8:         $y_{new}\ ,\ P_{new} \leftarrow \mathbf{F}(X'_j)$
9:         if $y_{new} \neq y_{orig}$ then
10:            $\mathcal{X_{ADV}} \leftarrow X'_j$
11:            return $\mathcal{X_{ADV}}$
12:        if $P_{new} < P_{best}$ then
13:            $P_{best} \leftarrow P_{new}$
14:            $\mathcal{X_{ADV}} \leftarrow X'_j$
15: return $\mathcal{X}_{ADV}$
```

4.3 Experiments

4.3.1 Datasets and Target Models

We use *IMDB* — A document level sentiment classification dataset for movie reviews [96] and *Yelp Reviews* — A restaurant review dataset [94], for classification task. We use *MultiNLI* — A natural language inference dataset [98] for entailment task.

Dataset	**Train**	**Test**	**Classes**	**Avg. Len**
IMDB	12K	12K	2	215
Yelp	560K	18K	2	152
MultiNLI	12K	4K	3	10

Table 4.1: Statistics of all datasets

4.3.2 Models

Target Models: We attacked WordLSTM [10] and BERT-base-uncased [19] for evaluating our attack strategy on text classification and entailment tasks. For WordLSTM a single layer bi-directional LSTM with 150 hidden units, 200 dimensional GloVe [107] vectors and a dropout of 0.3 were used.
Ranking Models: We used Hierarchical Attention Networks (HAN) and Decompose Attention Model (DA) for classification and entailment tasks respectively. For training HA, we used 200 dimensional word2vec embeddings and 50 dimensinal GRU cells. We used 100 dimensional word context vectors initialized randomly. We trained the model with a batch size of 64, learning rate of 0.0001, dropout of 0.3 and momentum of 0.9. For DA, a 2 layer LSTM with 200 neurons and 200 dimensional glove embeddings are used. A batch size of 32, learning rate of 0.025, dropout of 0.2 are used. All the hyper-parameters were tuned on the 20% validation set of each dataset.

4.3.3 Search Spaces and Baselines

We compare our search method with four baselines across three different search spaces. Also, while comparing our results with each baselines we use the same search space as used in that baseline paper. The details of search spaces are shown in Table 4.2.
PSO: [44] It uses particle swarm optimization algorithm as a search method and uses HowNet [75] as search space.
TextFooler: [43] It finds important words and replace them with synonyms from counter-fitted embeddings [50].
PWWS: [42] It takes word saliency score into account to rank words and uses WordNet [49] for syn-

onym substitution.
Genetic Attack: It crafts examples using a population based algorithm [40].

Baseline	**Transformation**	**Constraint**
Genetic Attack	Counter-fitted	Word similarity, LM score
TextFooler	embeddings	USE = 0.84, POS consistency
PSO	HowNet	USE = 0.84, POS consistency
PWWS	WordNet	POS consistency

Table 4.2: Baselines and their search spaces

Model	**Attack**	**IMDB**		**Yelp**		**MNLI**	
		Qrs	**Suc%**	**Qrs**	**Suc%**	**Qrs**	**Suc%**
BERT	PSO	81350.6	**99.0**	73306.6	**93.2**	4678.5	**57.97**
	Ours	**737**	97.4	**554.2**	91.6	**97.2**	56.1
LSTM	PSO	52008.7	**99.5**	43671.7	**95.4**	2763.3	**67.8**
	Ours	**438.1**	**99.5**	**357.6**	94.75	**79.8**	66.4

(a) Comparison with PSO.

Model	**Attack**	**IMDB**		**Yelp**		**MNLI**	
		Qrs	**Suc%**	**Qrs**	**Suc%**	**Qrs**	**Suc%**
BERT	Gen	7944.8	66.3	6078.1	**85.0**	1546.8	**83.8**
	Ours	**378.6**	**71.1**	**273.7**	84.4	**43.4**	81.9
LSTM	Gen	3606.9	97.2	5003.4	**96.0**	894.5	**87.8**
	Ours	**224**	**98.5**	**140.7**	95.4	**39.9**	86.4

(b) Comparison with Genetic Attack.

Model	**Attack**	**IMDB**		**Yelp**		**MNLI**	
		Qrs	**Suc%**	**Qrs**	**Suc%**	**Qrs**	**Suc%**
BERT	PWWS	1583.9	**97.5**	1013.7	**93.8**	190	**96.8**
	Ours	**562.9**	96.4	**366.2**	92.6	**66.1**	95.1
LSTM	PWWS	1429.2	**100.0**	900.0	**99.1**	160.2	**98.8**
	Ours	**473.8**	**100.0**	**236.3**	**99.1**	**60.1**	98.1

(c) Comparison with PWWS.

Model	**Attack**	**IMDB**		**Yelp**		**MNLI**	
		Qrs	**Suc%**	**Qrs**	**Suc%**	**Qrs**	**Suc%**
BERT	TF	1130.4	**98.8**	809.9	**94.6**	113	85.9
	Ours	**750**	98.4	**545.5**	93.2	**100**	**86.2**
LSTM	TF	544	**100.0**	449.4	**100.0**	105	95.9
	Ours	**330**	**100.0**	**323.7**	**100.0**	**88**	**96.2**

(d) Comparison with TextFooler (TF).

Table 4.3: Result comparison. Succ% is the attack success rate and Qrs is the average query count. Note as each baseline uses a different search space, our method will yield different results when comparing with each baseline.

4.3.4 Experimental Settings

In a black box setting, the attacker has no access to the training data of the target model. Therefore, we made sure to train the attention models on a different dataset. For attacking the target model trained on IMDB, we trained our attention model on the Yelp Reviews and vice-versa. For entailment, we trained the attention model on SNLI [97] and attacked the target model trained on MNLI. Following [43],

Model	Attack	IMDB		Yelp		MNLI	
		Pert%	I%	Pert%	I%	Pert%	I%
BERT	PSO	4.5	0.20	10.8	0.30	8.0	3.5
	Ours	**4.2**	**0.10**	**7.8**	**0.15**	**7.1**	**3.3**
LSTM	PSO	2.2	0.15	7.7	0.27	**6.7**	**1.27**
	Ours	**2.0**	**0.11**	**4.9**	**0.15**	6.8	1.3

(a) Comparison with PSO.

Model	Attack	IMDB		Yelp		MNLI	
		Pert%	I%	Pert%	I%	Pert%	I%
BERT	Gen	**6.5**	1.04	11.6	1.5	**8.7**	**1.9**
	Ours	6.7	**1.02**	**10.5**	**1.49**	9.2	2.1
LSTM	Gen	4.1	0.62	8.6	1.3	7.7	2.5
	Ours	**3.19**	**0.56**	**6.2**	**1.05**	**8.2**	**2.1**

(b) Comparison with Genetic Attack.

Model	Attack	IMDB		Yelp		MNLI	
		Pert%	I%	Pert%	I%	Pert%	I%
BERT	PWWS	**5.2**	**0.74**	**7.3**	**1.5**	**7.1**	1.71
	Ours	7.5	0.9	9.9	1.9	9.6	**1.48**
LSTM	PWWS	2.3	**0.3**	**4.8**	**1.29**	**6.6**	**1.5**
	Ours	**1.9**	0.4	5.5	**1.29**	7.8	2.1

(c) Comparison with PWWS.

Model	Attack	IMDB		Yelp		MNLI	
		Pert%	I%	Pert%	I%	Pert%	I%
BERT	TF	9.0	1.21	5.2	**1.1**	11.6	**1.23**
	Ours	**6.9**	**0.9**	**6.6**	1.2	**11.4**	1.41
LSTM	TF	**2.2**	2.3	5.7	2.06	**9.8**	1.7
	Ours	2.4	**1.5**	**5.3**	**1.5**	10.1	**1.4**

(d) Comparison with TextFooler (TF).

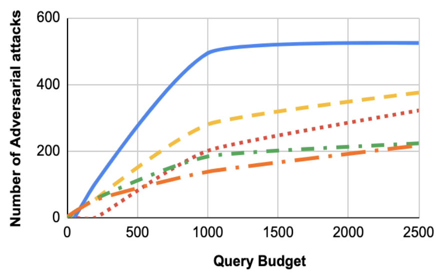
Table 4.4: Result comparison. Pert% is the perturbation and I% is the average grammatical error increase.

the target models are attacked on same 1000 samples, sampled from the test set of each dataset. The same set of samples are used across all baselines when evaluating on a single dataset. For entailment task we only perturb the premise and leave the hypothesis unchanged. We used spacy for POS tagging and filtered out stop words using NLTK. We used Universal Sentence Encoder [101] to encode the perturbed inputs while performing LSH. The hyperparameters d and L are tuned on the validation set (10% of each dataset). Additional details regarding hyperparameter tuning and attention models can be found in appendix.

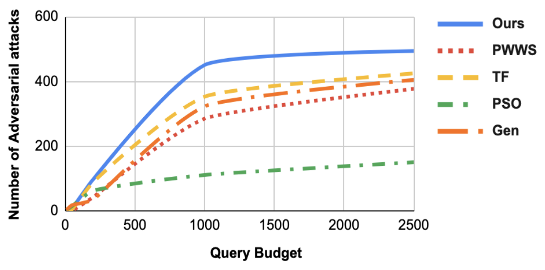
4.3.5 Evaluation Metrics

We use (1) *attack success rate* – the ratio of the successful attacks to total number of attacks, (2) *query count* – the number of queries, (3) *perturbation rate* – the percentage of words substituted in an input and (4) *grammatical correctness* – the average grammatical error increase rate (calculated using Language-Tool[1]) to verify the quality of generated adversarial examples. For all the metrics except attack success rate, lower the value better the result. Also, for all metrics, we report the average score across all the generated adversarial examples on each dataset. Further, we also conducted human evaluation to assess the quality of generated adversarial examples.

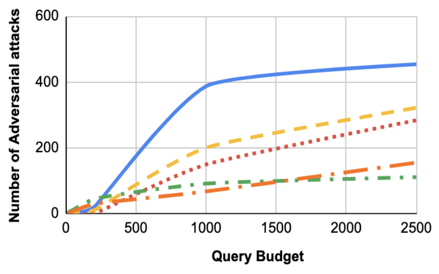
[1] https://languagetool.org/



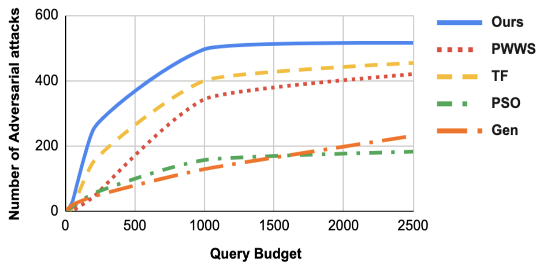
(a) Adv. samples generated against BERT on IMDB



(b) Adv. samples generated against BERT on Yelp



(c) Adv. samples generated against LSTM on IMDB



(d) Adv. samples generated against LSTM on Yelp

Figure 4.2: Comparison of the number of adversarial samples generated by varying the query budget L.

Dataset	**Random**			**Only Attention**			**Only LSH**			**Both LSH and Attention**		
	Suc%	Pert%	Qrs	Suc%	Pert%	Qrs	Suc%	Pert%	Qrs	Suc%	Pert%	Qrs
IMDB	90.5	13.3	507.9	94.0	9.3	851.3	95.3	8.0	694.9	**96.4**	**7.5**	**562.9**
Yelp	87.3	15.0	305.9	91.0	11.0	550.0	90.2	10.2	475.2	**92.6**	**9.8**	**366.2**
MNLI	88.8	14.3	60.1	92.4	11.7	121.2	94.3	10.1	100.1	**95.1**	**9.6**	**66.1**

Table 4.5: Ablation Study of attention mechanism and LSH on PWWS search space.

4.3.6 Results

Table 4.3 and 4.5 shows the comparison of our proposed method with each baseline across all evaluation metrics. On an average across all baselines, datasets and target models we are able to reduce the query count by 75%. The PSO and Genetic attack takes atleast $50x$ and $20x$ more queries respectively than our attack strategy. Also, when compared to PWWS and TF we are able to reduce the query count by atleast 65% and 33% respectively while compromising the success rate by less than 2.0%. The perturbation rate and the grammatical correctness is also within 1% of the best baseline. In comparison to PSO and Genetic Attack, we are able to achieve even a lower perturbation rate and grammatical error rate with much lesser queries across some datasets and target models. Similarly, our attack outperforms TextFooler almost on all evaluation metrics. The runtime comparison is provided in table 4.6 and the anecdotes from generated adversarial text are provided in the tables 4.9 and 4.10.

Attack	**Runtime**
PSO	72 hrs
Genetic Attck	10 hrs
PWWS	3 hrs
TextFooler	2.5 hrs
Ours	1 hrs

Table 4.6: Runtime comparison while attacking BERT trained on Yelp dataset on a set of 500 samples across WordNet search space.

4.4 Ablation Study

We study the effectiveness of attention and LSH component in our method by doing a three way ablation. We observe the change in success rate, perturbation rate and queries when both or either one of the two ranking components are removed.

No LSH and attention: First, we remove both the attention and LSH scoring steps and rank the words in random order. Table 4.5 shows the results obtained on BERT across all three datasets. On an average the attack success rate drops by 7%, the perturbation rate increases drastically by 6%. This shows that although the query count reduces, substituting words in random order degrades the quality of generated adversarial examples and is not effective for attacking target models.

Attention and no LSH: We remove the LSH component of our ranking step and rank words based upon only the attention scores obtained from the attention model. Table 4.5 shows the results on BERT across all datasets. On an average the attack success rate drops by 2.5%, the perturbation rate increases by 3% and the query increases by 37%. Therefore, LSH reduces the queries significantly by eliminating near duplicates in search space.

LSH and no Attention: We remove the attention component and rank words using only LSH. Results in Table 4.5 shows that on an average, without attention scoring the attack success rate drops by 2%, the perturbation rate increases by 0.5% and the query increases by 20%. Therefore, attention is important as it not only reduces queries but also enables the ranking method to prioritize important words required in target model prediction.

With LSH and Attention: Finally in Table 4.5 we observe that, using both LSH and attention in our ranking our attack has a much better success rate, a lower perturbation rate in much lesser queries. This shows that both the components are necessary to do well across all evaluation metrics.
We obtained similar results on LSTM when evaluating across different datasets and search spaces.

4.5 Quantitative Analysis

4.5.1 Limited Query Setting

In this setting, the attacker has a fixed query budget L, and can generate an attack in L queries or less. To demonstrate the efficacy of our attack under this constraint, we vary the query budget L and observe the attack success rate on BERT and LSTM across IMDB and Yelp datasets. We vary the query budget from 0 to 2500 and observe how many adversarial examples can be generated successfully on a test set of 500 samples. We keep the search space same (used in PWWS) across all the search methods. The results in Figure 4.6 shows that with a query budget of 1000, our approach generates atleast 200 (44.4%) more adversarial samples against both BERT and LSTM on IMDB when compared to the best baseline. Similarly, on Yelp our method generates atleast 100 (25%) more adversarial samples on BERT and LSTM when compared to the best baseline. This analysis shows that our attack has a much higher success rate in a limited query setting, thus making it extremely useful for real world applications.

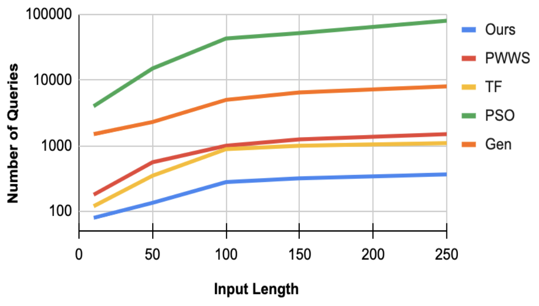


Figure 4.3: Queries taken vs number of words in input

Transfer	**Accurracy**	**IMDB**	**MNLI**
BERT $\rightarrow$ LSTM	Original	90.9	85.0
	Transferred	**72.9**	**60.6**
LSTM $\rightarrow$ BERT	Original	88.0	70.1
	Transferred	**67.7**	**62.1**

Table 4.7: Transferability on IMDB and MNLI datasets

4.5.2 Input Length

To demonstrate that how our strategy scales with change in the input length (number of words in the input) compared to other baselines, we attacked BERT on Yelp. We selected inputs having number of words in the range of 10 to 250 and observed the number of queries taken by each attack method. Results in figure 4.3 shows that our attack takes the least number of queries across all input lengths. Further, our attack scales much better on longer inputs (> 250 words) as it is $2x$ faster than PWWS and TextFooler, $13x$ faster than Genetic attack and $133x$ faster than PSO.

4.5.3 Transferability

An adversarial example is said to be *transferable*, if it is generated against one particular target model but is able to fool other target models as well. We implemented transferability on IMDB and MNLI datasets across two target models. The results are shown in Table 4.7. Our transferred examples dropped the accuracy of other target models on an average by 16%.

4.5.4 Adversarial Training

We randomly sample 10% samples from the training dataset of MNLI and IMDB and generate adversarial examples using our proposed strategy. We augmented the training data with the generated adversarial examples and re-trained BERT on IMDB and MNLI tasks. We then again attacked BERT with our proposed strategy and observed the changes. The results in Figure 4.4 shows that as we add more adversarial examples to the training set, the model becomes more robust to attacks. The after attack accuracy and perturbation rate increased by 35% and 17% respectively and required higher queries to attack.

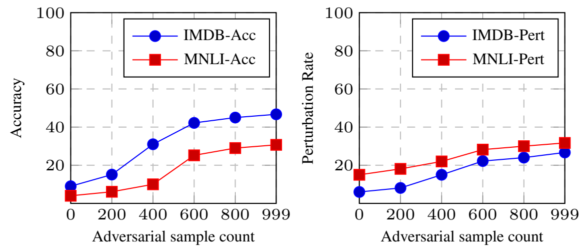


Figure 4.4: Increase in after attack accuracy and perturbation rate as more adversarial samples are augmented.

4.6 Qualitative Analysis

4.6.1 Human Evaluation

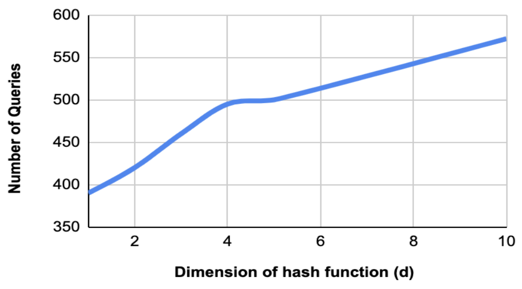
We verified the quality of generated adversarial samples via human based evaluation as well. We asked the evaluators to classify the adversarial examples and score them in terms of grammatical correctness (score out of 5) as well as its semantic similarity compared to the original text. We randomly sampled 25% of original instances and their corresponding adversarial examples generated on BERT for IMDB and MNLI datasets on PWWS search space. The actual class labels of adversarial examples were kept hidden and the human judges were asked to classify them. We also asked the human judges to evaluate each sample for its semantic similarity and assign a score of 0, 0.5 or 1 based on how well the adversarial examples were able to retain the meaning of their original counterparts. We also asked them to score each example in the range 1 to 5 for grammatical correctness. Each adversarial example was evaluated by 3 human evaluators and the scores obtained were averaged out. The outcome is in Table 4.8.

Evaluation criteria	**IMDB**	**MNLI**
Classification result	94%	91%
Grammatical Correctness	4.32	4.12
Semantic Similarity	0.92	0.88

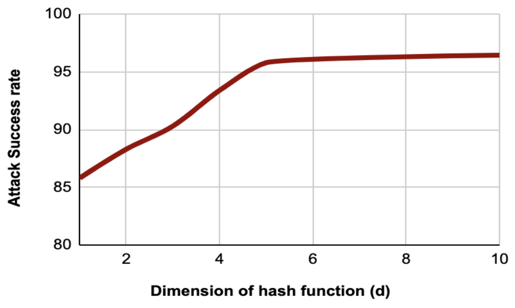
Table 4.8: Demonstrates scores given by evaluators

4.6.2 Hyperparameter Study

Figure 4.5a and 4.5b shows the variation of attack success rate and queries taken to attack the target model as d increases. With increase in d the number of collisions decreases and therefore the number of buckets obtained $\mathcal{K}$ increases. This increases the overall query count. Also, with increase in d the

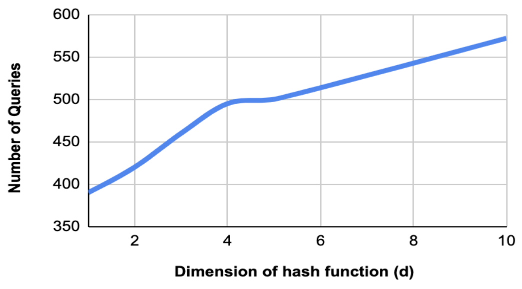


(a) Queries vs the dimension d of hash function

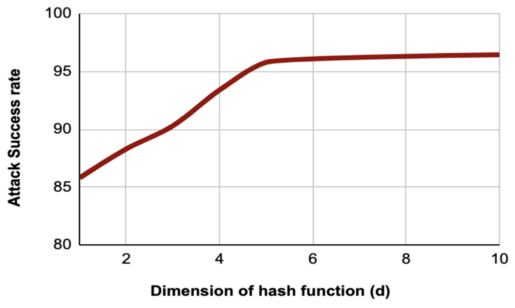


(b) Success rate vs dimension d of hash function

Figure 4.5: Comparison of success rate and queries required by changing the dimension d.



(a) Queries vs rounds of hashing L



(b) Success rate vs rounds of hashing L

Figure 4.6: Comparison of success rate and queries required by changing the rounds of hashing L.

success rate first increases and then remains unchanged. Therefore, we use $d = 5$ because after that the success rate is almost the same but the query count increases drastically. Figure 4.6a and 4.6b shows the variation of attack success rate and queries taken to attack the target model as the rounds of hashing L increases. Conducting multiple rounds of hashing reduces the probability that that similar perturbed text inputs are mapped to different buckets. We choose $L = 15$ as after it the attack success rate and the queries remain almost unchanged. The values of d, L are kept same across all datasets and target models.

Examples	**Prediction**
The movie has an excellent screenplay (the situation is credible, the action has pace), **first-class** [**fantabulous**] direction and acting (especially the 3 leading actors but the others as well -including the mobster, who does not seem to be a professional actor). I **wish** [**want**] the movie, the director and the actors success.	**Positive** → **Negative**
Let me start by saying I don't recall laughing once during this comedy. From the opening scene, our protagonist Solo (Giovanni Ribisi) shows himself to be a self-absorbed, feeble, and neurotic loser completely unable to cope with the smallest responsibilities such as balancing a checkbook, keeping his word, or forming a coherent thought. I guess we're supposed to be drawn to his fragile vulnerability and cheer him on through the process of clawing his way out of a deep depression. I actually **wanted** [**treasured**] him to get his kneecaps busted at one point. The dog was not a character in the film. It was simply a prop to be used, neglected, scorned, abused, coveted and disposed of on a whim. So be warned.	**Negative** → **Postive**
Local-international **gathering** [**assembly**] **spot** [**stain**] since the 1940s. One of the coolest pubs on the planet. Make new friends from all over the world, with some of the **best** [**skilful**] regional and imported beer selections in town.	**Postive** → **Negative**
This film is strange, even for a silent movie. Essentially, it follows the adventures about a engineer in post-revolutionary Russia who daydreams about going to Mars. In this movie, it seems like the producers KNOW the Communists have truly screwed up the country, but also seems to want to make it look like they've accomplished something good. Then we get to the "Martian" scenes, where everyone on Mars wears goofy hats. They have a revolution after being inspired by the Earth Men, but are quickly betrayed by the Queen who sides with them. Except it's all a dream, or is it. (And given that the Russian Revolution eventually lead to the Stalin dictatorship, it makes you wonder if it was all allegory.) **Now** [**Nowdays**], I've seen GOOD Russian cinema. For instance, Eisenstein's Battleship Potemkin is a good movie. This is just, well, silly.	**Negative** → **Postive**

Table 4.9: Demonstrates adversarial examples generated after attacking BERT on classification task. The actual word is highlighted green and substituted word is in square brackets colored red. Prediction shows before and after labels marked green and red respectively.

Examples	**Prediction**
Premise: If we travel for 90 minutes, we could **arrive** **[reach]** arrive at larger ski resorts. **Hypothesis**: Larger ski resorts are 90 minutes away.	**Entailment** → **Neutral**
Premise: I should put it in this **way** **[manner]**. **Hypothesis**: I'm not explaining it.	**Entailment** → **Neutral**
Premise: **Basically** **[Crucially]**, to sell myself. **Hypothesis**: Selling myself is a very important thing.	**Contradict** → **Neutral**
Premise: **June** **[April]** 21, 1995, provides the specific requirements for assessing and reporting on controls. **Hypothesis**: There are specific requirements for assessment of legal services.	**Contradict** → **Entail**

Table 4.10: Demonstrates adversarial examples generated after attacking BERT on entailment task. The actual word is highlighted green and substituted word is in square brackets colored red. Prediction shows before and after labels marked green and red respectively.

Chapter 5

A Context Aware Approach for Generating Natural Language Attacks

5.1 Introduction

In this chapter [108], we study an important task of attacking natural language processing models in a black box setting. We propose an attack strategy that crafts semantically similar adversarial examples on text classification and entailment tasks. Our proposed attack finds candidate words by considering the information of both the original word and its surrounding context. It jointly leverages masked language modelling and next sentence prediction for context understanding. In comparison to attacks proposed in prior literature, we are able to generate high quality adversarial examples that do significantly better both in terms of success rate and word perturbation percentage.

Existing black box attacks in NLP [43, 42] generate adversarial examples using a two step process. (1) It finds important words which highly impact the classification score of the target model. (2) It replaces those words with a synonym generated either by using a lexical database such as WordNet or from the nearest neighbour in the counter-fitted embedding space. Though this method is effective in generating adversarial attacks, it has the following drawbacks. (1) It takes only the word level similarity to find synonyms but does not consider the overall context surrounding the replaced word thus generating out of context synonyms which degrades the overall semantics. (2) It yields unnatural and complex replacements which results in grammatically incorrect and non-fluent adversarial examples.

In this chapter, we propose a black box attack that crafts high quality adversarial examples on text classification and entailment tasks. For each word to be replaced, our proposed attack generates candidate words using the influence of both the original word (to be replaced) as well as its surrounding context. The generated candidates (1) fit well within the sentence thus retaining the overall semantics of the sentence, (2) have the same meaning as that of the original word and (3) are grammatically correct and fluent. We use BERT to generate candidates for each word to be replaced in the input. BERT is a masked language model trained on masked language modeling (MLM) — predicts the $[MASK]$ word using both left and right context and next sentence prediction (NSP) — predicts the next sentence given previous sentences separated by the $[SEP]$ token. For each word to be replaced, we substitute that word

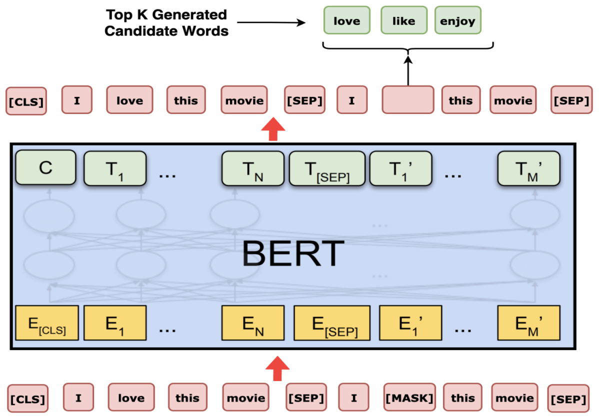


Figure 5.1: Overview of proposed strategy

with a $[MASK]$ token to get a masked input. Then, we leverage the NSP and feed both the original input as well as the masked input separated by the $[SEP]$ to BERT.

5.2 Proposed Approach

Given a target model $\mathbf{F} : \mathcal{X} \to \mathcal{Y}$, that classifies an input text $\mathcal{X}$ to a set of class labels $\mathcal{Y}$. Our goal is to generate an adversarial text sequence $\mathcal{X}_{\mathcal{ADV}}$ that is misclassified by $\mathbf{F}$ i.e. $\mathbf{F}(\mathcal{X}) \neq \mathbf{F}(\mathcal{X}_{\mathcal{ADV}})$ and is semantically similar to $\mathcal{X}$. We solve this problem using the following two steps:

Word Ranking: Given an input text $\mathcal{X} = \{x_1, x_2...x_n\}$, this step assigns high scores to words which significantly impact the final prediction of $\mathbf{F}$. To score a word x_i, this step removes x_i from the input and queries $\mathbf{F}$ to observe the change in the classification score of the target class. This process is repeated for all of the words in $\mathcal{X}$. All the words are then sorted in descending order based on their score.

Word Substitution: Given the word x_i and $\mathcal{M}$ the BERT masked language model, this step finds a word replacement for x_i using the information of both the original word x_i and its surrounding context. It consists of the following steps:

(1) Candidate Generation: It replaces x_i with a $[MASK]$ token to get a masked input $\mathcal{X}_i = \{x_1..x_{i-1}, [MASK], x_{i+1}..x_n\}$. But, simply masking a word and using $\mathcal{M}$ to predict the masked word might generate candidate words that are not synonyms of x_i. For example, consider the sentence "I love this movie", if the word "love" is masked than $\mathcal{M}$ might predict words like "hate" and "dislike" with high probability which in turn will alter the overall semantics of the input. Therefore, to capture the influence of the original word x_i, we leverage NSP and feed both the original input $\mathcal{X}$ as well as the masked input $\mathcal{X}_i$ separated by a $[SEP]$ token to $\mathcal{M}$. Feeding the sentence pair ($\mathcal{X}$,$\mathcal{X}_i$) generates candidate words which not only fits the given context but also retain the meaning of the original word x_i. We take the top $\mathcal{K}$ predicted words for each word x_i for substitution. Further, to ensure that the

Model	Orig%	Attack	Acc%	Pert%	I%
BERT (IMDB)	90.9	TF	13.6	6.1	0.5
		Ours	**9.5**	**4.2**	**0.38**
LSTM (IMDB)	89.8	GA	3.0	14.7	0.78
		PWWS	2.0	3.38	0.46
		TF	0.3	5.1	0.53
		Ours	**0.3**	**3.2**	**0.32**
BERT (MR)	86.5	TF	11.5	16.7	1.26
		Ours	**10.7**	**16.3**	**0.7**
LSTM (MR)	80.0	TF	3.1	14.9	1.04
		PWWS	3.7	14.38	0.86
		Ours	**2.1**	**14.0**	**0.7**
BERT (SNLI)	89.1	TF	4.0	**18.5**	9.7
		Ours	**3.6**	18.9	**1.9**
LSTM (SNLI)	84.0	GA	30	23.3	9.9
		TF	3.5	18.0	7.7
		Ours	**3.0**	**17.0**	**2.3**

Table 5.1: Result comparison. Orig% is the original accuracy, Acc% is the after attack accuracy, Pert% is the average perturbation rate, I% is the grammatical error increase rate.

generated candidates are highly similar to the original word x_i we apply counter-fitting and filter out candidates with cosine similarity less then the threshold.

(2) Final candidate selection: Each remaining candidate is substituted for x_i in $\mathcal{X}$ which results in a perturbed text sequence $\mathcal{X}_i^{'}$. Each perturbed sequence is checked for semantic similarity with $\mathcal{X}$ and sequences with threshold less than λ are filtered out. The remaining perturbed sequences are fed to the target model **F**. The perturbed text sequence which alters the final prediction of the target model is selected as the final adversarial example. In case the prediction does not change then the perturbed sequence which results in the least confidence score of the target class is selected and the above steps are repeated on the chosen perturbed sequence for the next ranked word.

Experiments We evaluate our proposed attack strategy across two state-of-the art target models — WordLSTM, BERT-base-uncased on three benchmark datasets — IMDB, MR and SNLI. Note that the BERT masked language model $\mathcal{M}$ used to generate candidates is different from the BERT target model as the latter is fine tuned for each dataset. We use *after attack accuracy* — accuracy achieved on the generated adversarial examples, *perturbation rate* — the percentage of words substituted in an input and *grammatical correctness* — average grammatical error increase rate as our evaluation metrics. For all these three metrics, the lower the value, better the attack result. We compare our attack with three

Type	**Examples**
Orig.	Vile and tacky are best adjectives to describe it.
TF	Vile and tacky are best **qualifier** to describe it.
Ours	Vile and tacky are best **words** to describe it.
Orig.	This film is a portrait of grace in this world.
TF	This film is a **spitting** of grace in this **universe**.
Ours	This film is a **sketch** of **beauty** in this world.

Table 5.2: Demonstrates adversarial examples generated on BERT by TF and our attack. Substituted words are in bold.

state-of-the-art baselines, (1) TextFooler (TF) [43] (2) PWWS [42] (3) Genetic Attack (GA) [40]. We attack the target model on a test set of 1000 examples, sampled from the test set of each dataset. To ensure fair comparison, these 1000 examples are the same set of samples as used in TF and GA. We use Universal Sequence Encoder (USE) to measure the semantic similarity between the original and the adversarial example. We set the semantic similarity threshold (λ) to 0.8 for classification and 0.6 for entailment tasks. We take 30 as the context window, and set $\mathcal{K}$ to 50.

Attack Performance As shown in Table 5.1, we are able to achieve more than 90% success rate across all target models and datasets. In comparison to TextFooler, on average our attack reduces the after attack accuracy and perturbation rate by 15% and 14% respectively across all target models and datasets. Further it reduces the grammatical error rate by at least 30%. Our attack also significantly outperforms other baselines on all evaluation metrics.

Chapter 6

Adversarial Examples for Evaluating Math Word Problem Solvers

6.1 Introduction

A Math Word Problem (MWP) consists of a natural language text which describes a world state involving some known and unknown quantities. The task is to parse the text and generate equations that can help find the value of unknown quantities. Solving MWP's is challenging because apart from understanding the text, the model needs to identify the variables involved, understand the sequence of events, and associate the numerical quantities with their entities to generate mathematical equations. An example of a simple MWP is shown in Table 6.1.

In recent years, solving MWPs has become a problem of central attraction in the NLP community. There are a wide variety of MWPs ranging from simple linear equations in one variable [109, 110] to complex problems that require solving a system of equations [111, 112]. In this paper, we consider simple MWPs which can be solved by a linear equation in one variable.

Existing MWP solvers can be categorized into statistical learning based [5, 6] and deep learning based solvers . However, recent deep learning based approaches [113, 88, 114] have established their superiority over statistical learning based solvers. Here, we will briefly review some recent MWP solvers. Initially, [113] modelled the task of MWP as a sequence to sequence task and utilized Recurrent Neural Nets (RNNs) to learn problem representations. Building upon this, [86] focused on learning representations for mathematical operators and numbers, [88, 87] utilized tree structure to develop decoders for MWP solvers. More recently, to learn accurate relationship between numerical quantities and their attributes [114] modelled encoder as a graph structure.

All such MWP solvers have achieved high performance on benchmark datasets. However, the extent to which these solvers truly understand language and numbers remains unclear. Prior works on various NLP tasks have shown that Deep Neural Networks (DNNs) attend to superficial cues to achieve high performance on benchmark datasets. Recently, [115] proposed a challenge test set called SVAMP which demonstrate that existing MWP solvers rely on shallow heuristics to achieve high performance. Instead of relying on standard accuracy metrics, many works have used adversarial examples [30, 64] to evaluate the robustness of neural NLP models. Adversarial examples are generated by making small changes to

Original Problem

Text: Tim has 5 books. Mike has 7 books.

How many books do they have together?

Equation: X = 5+7

Question Reordering

Text: How many books do they have together

given that Tim has 5 books and Mike has 7 books.

Equation: X = 5*7

Sentence Paraphrasing

Text: Tim has got 5 books. There are 7 books in

Mike's possession. How many books do they have?

Equation: X = 5*5

Table 6.1: A MWP and generated adversarial examples by our methods. Red and blue color denote the subject and the entity respectively of numerical values.

the original input such that the adversarial example is (1) semantically similar to the original input, (2) is grammatically correct and fluent and (3) deceives the DNNs to generate an incorrect prediction.

In [55] authors crafted adversarial attacks to test the robustness of QA systems. Prior works in [63, 116] uses adversarial examples to show deficiencies of NLI models. Similarly, [60, 58] uses adversarial examples to develop robust dialogue and neural machine translation models. Recently, there has been a plethora of work [36, 40, 52, 61, 108, 62] to evaluate text classification systems against adversarial examples. Although adversarial examples are commonly used for various NLP tasks, there has been no work that uses adversarial examples to evaluate MWP solvers. In this chapter [117], we bridge this gap and evaluate the robustness of state-of-the-art MWP solvers against adversarial examples.

Generating adversarial attacks for MWP is a challenging task as apart from preserving textual semantics, numerical value also needs to be preserved. The text should make mathematical sense, and the sequence of events must be maintained such that humans generate the same equations from the problem text. Standard adversarial generation techniques like synonym replacement [40] are not suitable for MWP as the fluency of the problem statement is not preserved. Similarly, paraphrasing techniques like back-translation [118] are not ideal as they generate syntactically uncontrolled examples.

We propose two methods to generate adversarial examples on MWP solvers, (1) Question Reordering — It transforms the problem text by moving the question part to the beginning of the problem and (2) Sentence Paraphrasing — It paraphrases each sentence in the problem such that the semantic meaning and the numeric information remains unchanged. Our results demonstrate that current solvers are not robust against adversarial examples as they are sensitive to minor variations in the input. We hope that our insights will inspire future work to develop more robust MWP solvers. Our contributions are as follows:

1. To the best of our knowledge, this is the first work that evaluates the robustness of MWP solvers against adversarial attacks. We propose two methods to generate adversarial examples on three MWP solvers across two benchmark datasets.

2. On average, the generated adversarial examples are able to reduce the accuracy of MWP solvers by over 40%. Further, we experiment with different type of input embeddings and perform adversarial training using our proposed methods. We also conducted human evaluation to ensure that the generated adversarial examples are valid, semantically similar and grammatically correct.

6.2 Proposed Approach

6.2.1 Problem Definition

A MWP is defined as an input of n tokens, $\mathcal{P} = \{w_1, w_2..w_n\}$ where each token w_i is either a numeric value or a word from a natural language. The goal is to generate a valid mathematical equation $\mathcal{E}$ from $\mathcal{P}$ such that the equation consists of numbers from $\mathcal{P}$, desired numerical constants and mathematical operators from the set $\{/, *, +, -\}$. The above problem can also be expressed as $\mathcal{P} = \{S_1, S_2..S_k, Q\}$ where Q is the question, $\{S_1, S_2..S_k\}$ are the sentences constituting the problem description.
Let **F** : $\mathcal{P} \to \mathcal{E}$ be a MWP solver where $\mathcal{E}$ is the solution equation to problem $\mathcal{P}$. Our goal is to craft an adversarial text input $\mathcal{P}^*$ from the original input $\mathcal{P}$ such that the generated sequence is (1) semantically similar to the original input, (2) preserves sequence of events in the problem, (3) preserve numerical values and (4) makes the MWP solver **F** to generate an incorrect equation $\mathcal{E}^*$ for the unknown variable. We assume a black-box setting in which we have no access to the parameters, architecture or training data of the MWP solver. We only have access to the input text and equations generated by the solver.

6.2.2 Question Reordering

To examine whether existing MWP solvers are sensitive to the order of the question in the problem text, we moved the question Q at the start, followed by the rest of the problem description $\{S_1, S_2..S_k\}$. Formally, given the original input $\mathcal{P} = \{S_1, S_2...S_k, Q\}$ we transformed this to $\mathcal{P}^* = \{Q, S_1, S_2...S_k\}$. We keep the rest of the problem description $\{S_1, S_2..S_k\}$ unaltered. Also, to ensure that the generated problem text $\mathcal{P}^*$ is grammatically correct and fluent, we added phrases like ”Given that” or ”If” after the end of the question Q and before the start of the sentences $\{S_1, S_2..S_k\}$. An example of this is shown in

Table 1. We additionally, make use of co-reference resolution and named entity recognition[1] to replace pronouns with their co-referent links. Note that placing the question Q at the start rather than any other position ensures that the generated problem $\mathcal{P}^*$ has the same sequence of events as the original problem $\mathcal{P}$. Moreover, this method is better than randomly shuffling the sentences in $\mathcal{P}$ as it can change the sequence of events in the problem, resulting in a completely different equation.

6.2.3 Sentence Paraphrasing

To check whether MWP solvers generate different equations to semantically similar inputs, we generate paraphrases of each sentence in the problem text. Sentence Paraphrasing ensures that solvers do not generate equations based on keywords and specific patterns. Formally, given a problem statement $\mathcal{P}$ we obtain top m paraphrases for each sentence S_i as $\{S_{i,1}, S_{i,2}, ..., S_{i,m}\}$ and for question Q as $\{Q_{i,1}, Q_{i,2}, ..., Q_{i,m}\}$ by passing it through a paraphrasing model $\mathcal{M}$. For sentences with numerical values present in them, we need to ensure that each paraphrase candidate associates the numeric values with the same entity and subject as it is present in the original sentence S_i. To ensure this, we follow the approach used in [5] to segregate each sentence S_i into entities and its subject. These are collectively labeled as head entity $h_{i,orig}$ for the original sentence S_i and $h_{i,k}$ for the paraphrase candidates $S_{i,k}$. This methodology ensures that each numeric value is still associated correctly with its attributes even after paraphrasing. Paraphrased sentences that do not have matching head entities for any of the numeric values are filtered out. The remaining paraphrases of S_i and question Q are combined to generate all possible combinations of problem texts. The input combination for which the MWP solver generates an incorrect or invalid equation is selected as the final adversarial problem text $\mathcal{P}^*$. Sentence Paraphrasing generates inputs containing small linguistic variations and diverse keywords (more examples in appendix). Therefore, it is used to evaluate whether existing MWP solvers rely on specific keywords or patterns to generate equations. Algorithm 4 shows all the steps followed above to generate paraphrases.

6.3 Experiments

6.3.1 Datasets and Models

We evaluate the robustness of three state-of-the-art MWP solvers: (1) *Seq2Seq* [113] having an LSTM encoder and an attention based decoder. (2) *GTS* [88] having an LSTM encoder and a tree based decoder and (3) *Graph2tree* [114] consists of a both a tree based encoder and decoder. Many existing datasets are not suitable for our analysis as either they are in Chinese [113] or they have problems of higher complexities [111] . We conduct experiments across the two largest available English language

[1]https://spacy.io/

Algorithm 4 Sentence Paraphrasing

Input: Problem text $\mathcal{P}, \mathcal{M}$ is Paraphrase model

Output: Adversarial text $\mathcal{P}^*$

```
1: $\mathcal{P}^* \leftarrow \mathcal{P}$
2: $y_{orig} \leftarrow \mathbf{F}(\mathcal{P})$
3: for $S_i\ in\ \mathcal{P}$ do
4:     $C \leftarrow \mathcal{M}(S_i)$
5:     for $c_j\ in\ C$ do
6:         if $h_{i,orig} == h_{i,j}$ then
7:             $paraphrases.add(c_j)$
8:     $paraphrases.add(S_i)$
9:     $candidates.add(paraphrases)$
10: for $c_k$ in $Combinations(candidates)$ do
11:     $y_{adv} \leftarrow \mathbf{F}(c_k)$
12:     if $y_{adv} \neq y_{orig}$ then
13:         $\mathcal{P}^* \leftarrow c_k$
14:         end
```

datasets satisfying our requirements: (1) *MaWPS* [109] containing $2,373$ problems (2) *ASDIV-A* [110] containing $1,213$ problems. Both datasets have MWPs with linear equation in one variable.

6.3.2 Experimental Setup

We trained the three MWP solvers from scratch as implemented in baseline paper [113] on the above two datasets using 5-fold cross-validation as followed in [114]. The original accuracies obtained on the datasets are shown in Table 6.2. We used [119] to generate paraphrases of each sentence in the problem text. Same hyperparameter values were used as present in the original implementation of the paraphrase model. We conducted a human evaluation (Section 6.4.3) to verify the quality of generated adversarial examples. Further details are given in Appendix.

6.3.3 Implementation Details

For conducting our experiments we have used two Boston SYS-7048GR-TR nodes equipped with NVIDIA GeForce GTX 1080 Ti computational GPU's . The number of parameters ranged from 20M to 130M for different models. Hyper-parameter values were not modified, and we follow the recommendations of the respective models. We chose the number of candidate paraphrases m used in Algorithm 4 to be 7. Generating adversarial examples using Question Reordering took around 3 minutes on average for both MaWPS and ASDiv-A dataset. Sentence Paraphrasing took around 10 minutes on average for generation of adversarial examples on both the datasets. These experiments are not computation heavy as the generation technique is of linear order and number of examples are moderate.

6.3.4 Results

Table 6.2 shows the results of our proposed methods. On average, the generated adversarial examples can lower the accuracy of MWP solvers by over 40 percentage points. Across both datasets, *Graph2Tree*, the state-of-the-art MWP solver achieves only 34% and 24% accuracy on *Question Reordering* and *Sentence Paraphrasing* respectively. *Sentence Paraphrasing* is around 10 percentage points more successful in attacking MWP solvers than *Question Reordering*. These results verify our claim that current MWP solvers are sensitive to small variations in the input. Table 6.1 shows an MWP problem and its adversarial counterparts generated by our method and more examples are in the Table 6.5 and 6.6.

Dataset	Eval Type	Seq2Seq	GTS	Graph2Tree
MaWPS	Orig	53.0	82.6	83.7
	QR	**18.2**	**32.3**	**35.6**
	SP	**10.5**	**22.7**	**25.5**
ASDIV-A	Orig	54.5	71.4	77.4
	QR	**17.5**	**30.5**	**33.5**
	SP	**13.2**	**21.2**	**23.8**

Table 6.2: Results of MWP Solvers on adversarial examples. Orig is the original accuracy, QR is Question Reordering and SP is Sentence Paraphrasing.

6.4 Analysis

6.4.1 BERT Embeddings

We trained the solvers using pre-trained BERT embeddings and then generated adversarial examples against them using our proposed methods. Results obtained are shown in Table 6.3. We see that using BERT embeddings, the original accuracy of MWP solvers increases by 5 percentage points, and they are more robust than solvers trained from scratch. Specifically, these solvers do well against *Question*

Dataset	Eval Type	Seq2Seq	GTS	Graph2Tree
MaWPS	Adv (QR)	32.4	52.3	54.9
	Adv (SP)	27.6	40.7	42.3
	BERT (QR)	**45.3**	**63.0**	**65.6**
	BERT (SP)	**32.5**	**43.5**	**45.5**
ASDIV-A	Adv (QR)	34.5	48.4	54.8
	Adv (SP)	28.8	31.6	33.0
	BERT (QR)	**41.3**	**59.8**	**62.7**
	BERT (SP)	**30.6**	**40.0**	**42.6**

Table 6.3: Accuracy of MWP solvers with adversarial training on our proposed methods. Adv and BERT represent models trained from scratch and BERT embeddings respectively.

Reordering because of the contextualized nature of BERT embeddings, but for examples generated using *Sentence Paraphrasing* methods these models do not perform well. However, on average, our adversarial examples can lower the accuracy by 30 percentage points on both datasets.

6.4.2 Adversarial Training

To examine the robustness of MWP solvers against our attacks, we generated adversarial examples on the training set of both the datasets using our proposed methods and then augmented the training sets with the generated adversarial examples. We then retrained the MWP solvers and again attacked these solvers using our methods. Table 6.3 shows that the MWP solvers become more robust to attacks. Specifically, the solvers perform well against *Question Reordering* but are still deceived by *Sentence Paraphrasing*. Nevertheless, our proposed attack methods are still able to lower the accuracy of MWP solvers by 25 percentage points.

6.4.3 Human Evaluation

To verify the quality and the validity of the adversarial examples, we asked human evaluators (1) To check if the paraphrases will result in the same linear equation as that of the original problem, (2) Evaluate each adversarial example in the range 0 to 1 to check its semantic similarity with the original problem and (3) On a scale of 1 to 5 rate each adversarial example for its grammatical correctness. We also explicitly check for examples which do not satisfy our evaluation criteria and manually remove them from adversarial examples set. Three different human evaluators evaluate each sample, and the mean results are shown in Table 6.4.3.

Evaluation criteria	**MaWPS**	**ASDIV-A**
Same Linear Equation	85.7%	86.2%
Semantic Similarity	0.88	0.89
Grammatical Correctness	4.55	4.63

Table 6.4: Human Evaluation scores on datasets

Original Problem

Problem Statement : A teacher had 7 worksheets to grade . If she graded 3 , but then another 4 were turned in, how many worksheets would she have to grade ?

Predicted Equation : X = 7+3-4

Question Reordering

Problem Statement : How many worksheets would she have to grade given that a teacher had 7 worksheets to grade and if she graded 3 but then another 4 were turned in?

Predicted Equation : X = 7+3+4

Sentence Paraphrasing

Problem Statement : A teacher had her students work on 7 questions. 3 would be graded if she graded it. Then another 4 was turned in. How many things would she have to grade?

Predicted Equation : X = 7+3-4

Original Problem

Problem Statement : Gwen earned 20 points for each bag of cans she recycled . If she had 10 bags, but didn't recycle 3 of them , how many points would she have earned ?

Predicted Equation : X = (20*(10-3))

Question Reordering

Problem Statement : How many points would she have earned given that Gwen earned 20 points for each bag of cans she recycled and if she had 10 bags but didn't recycle 3 of them ?

Predicted Equation : X = 20*10-3

Sentence Paraphrasing

Problem Statement : Gwen earned 20 points for each bag of cans she recycled. She have 10 bags. She did not recycle 3 of them. How many points would she have gotten?

Predicted Equation : X = 20+10-3

Original Problem

Problem Statement : : Dennis has 12 pencils stored in boxes. If there are 3 boxes, how many pencils must go in each box?

Predicted Equation : X = 12/3

Question Reordering

Problem Statement : : If there are 3 boxes, how many pencils must go in each box given that Dennis has 12 pencils stored in boxes ?

Predicted Equation : X = 12/3

Sentence Paraphrasing

Problem Statement : Dennis has 12 pencils in boxes. There are 3 boxes. Find the number of pencils in each box?

Predicted Equation : X = 12-3

Table 6.5: Some instances of valid Adversarial Examples

Original Problem

Problem Statement : A trivia team had 10 members total. But during a game 2 members did not show up. If each member that did show up scored 3 points. How many points were scored?

Predicted Equation : X = (10-2)*3

Sentence Paraphrasing

Problem Statement : A team with 10 members had a lot of questions to answer. But during the game 2 members did not show up. 3 points were scored if each member showed up. How many points were scored?

Original Problem

Problem Statement : A tailor cut 15 of an inch off a skirt and 5 of an inch off a pair of pants . How much more did the tailor cut off the skirt than the pants ?

Predicted Equation : X = 15-5

Sentence Paraphrasing

Problem Statement : The 15 was cut by a tailor. There is a skirt and 5 of an inch off. There is a pair of pants. How much more did the tailor cut off the skirt than the pants?

Original Problem

Problem Statement : A vase can hold 10 flowers . If you had 5 carnations and 5 roses,
how many vases would you need to hold the flowers?

Predicted Equation : X = (5+5)/10

Sentence Paraphrasing

Problem Statement : 10 flowers can be held in a vase. If you had 5 and 5 roses. How many vases
do you need to hold the flowers.

Table 6.6: Some instances of invalid Adversarial Examples

Chapter 7

Conclusions

In this thesis, we introduced two different attack setting to evaluate NLP models — *hard label black box setting* and *limited query setting*. Adversarial attacks carried out in the above setting mimics closely the attacks that can be carried out in real world scenarios. Next, we evaluated the robustness of NLP models by proposing two novel attack methods that generate plausible and semantically similar adversarial examples in the above attack setting. Our first attack method is a decision-based attack utilizes population-based optimization algorithm to craft adversarial examples by observing only the topmost predicted label. In comparison to prior attack strategies, it achieves a higher success rate and lower perturbation rate that too in a highly restricted hard label black box setting. Our second method used attention scores and LSH to rank input words. This ranking mechanism is query efficient as it takes significantly lesser queries than prior attacks and works the best in limited query setting. Extensive experiments across multiple target models and datasets demonstrates the effectiveness of our proposed attack methods and showcase that DNNs are not robust when evaluated in a restricted setting that simulates the real world scenarios.

We benchmarked our results across the same search space used in prior attacks so as to ensure fair and consistent comparison. To improve the quality of generated adversarial examples, we propose an alternative method that uses masked language model to find candidate words by considering the information of both the original word and its surrounding context. Apart from this, we generated adversarial examples against various math word problem solvers and our results demonstrated that existing MWP solvers are not robust and do not truly understand language and its relation with numerical quantities.

Chapter 8

Future Work

Although our attack methods outperforms all baselines in two realistic setting, there are few drawbacks which we leave for future. First, the number of queries taken to generate adversarial examples in the hard label black box attack method is relatively high in comparison to baselines. Future attack methods could focus on reducing queries in a hard label balck box setting. Our query efficient attack uses word level scoring attention scores that can be extended to sentence level to further reduce the query count. Also, the attention scoring model used can be trained on different datasets to observe how the success rate and the query efficiency gets affected.

Most of the work on adversarial attacks in NLP are focused on proposing new attack methods only and there has been limited work in defending target models against adversarial attacks. Researchers should focus on proposing novel defense methods and also on evaluating existing attacks against various defense methods to compare the effectiveness of different attacks. Furthermore, many existing methods focus on evaluating models trained for specific tasks and do not pay heed to evaluate state of the art large language models against adversarial examples.

Our work on math word problem (MWP) solvers encourages the development of robust MWP solvers and techniques to generate adversarial math examples. We believe that the generation of quality MWP's will immensely help develop solvers that genuinely understand numbers and text in combination. Similar to [120], future works could focus on creating more such techniques for adversarial examples generation and making robust MWP solvers.

Related Publications

1. R. Maheshwary, S. Maheshwary, and V. Pudi, “Generating natural language attacks in a hard label black box setting,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 13525–13533, 2021

2. R. Maheshwary, S. Maheshwary, and V. Pudi, “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*, (Online and Punta Cana, Dominican Republic), pp. 8396–8409, Association for Computational Linguistics, Nov. 2021

3. R. Maheshwary, S. Maheshwary, and V. Pudi, “A context aware approach for generating natural language attacks,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 15839–15840, 2021

4. V. Kumar, R. Maheshwary, and V. Pudi, “Adversarial examples for evaluating math word problem solvers,” in *Findings of the Association for Computational Linguistics: EMNLP 2021*, (Punta Cana, Dominican Republic), pp. 2705–2712, Association for Computational Linguistics, Nov. 2021

5. V. Kumar, R. Maheshwary, and V. Pudi, “Practice makes a solver perfect: Data augmentation for math word problem solvers,” in *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, (Seattle, United States), pp. 4194–4206, Association for Computational Linguistics, July 2022

Bibliography

[1] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[2] Y. Freund, R. Schapire, and N. Abe, “A short introduction to boosting,” *Journal-Japanese Society For Artificial Intelligence*, vol. 14, no. 771-780, p. 1612, 1999.

[3] T. Chen and C. Guestrin, “Xgboost: A scalable tree boosting system,” in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.

[4] L. R. Rabiner, “A tutorial on hidden markov models and selected applications in speech recognition,” *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[5] M. J. Hosseini, H. Hajishirzi, O. Etzioni, and N. Kushman, “Learning to solve arithmetic word problems with verb categorization,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (Doha, Qatar), pp. 523–533, Association for Computational Linguistics, Oct. 2014.

[6] N. Kushman, Y. Artzi, L. Zettlemoyer, and R. Barzilay, “Learning to automatically solve algebra word problems,” in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (Baltimore, Maryland), pp. 271–281, Association for Computational Linguistics, June 2014.

[7] A. Kaul, S. Maheshwary, and V. Pudi, “Autolearn—automated feature generation and selection,” in *2017 IEEE International Conference on data mining (ICDM)*, pp. 217–226, IEEE, 2017.

[8] S. Maheshwary, A. Kaul, and V. Pudi, “Data driven feature learning,” 2017.

[9] S. Maheshwary, *Automated Feature Construction and Selection*. PhD thesis, International Institute of Information Technology Hyderabad, 2019.

[10] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[12] S. Maheshwary, S. Ganguly, and V. Pudi, “Deep secure: A fast and simple neural network based approach for user authentication and identification via keystroke dynamics,” in *IWAISe: First International Workshop on Artificial Intelligence in Security*, p. 59, 2017.

[13] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

[14] Y. Kim, “Convolutional neural networks for sentence classification,” *arXiv preprint arXiv:1408.5882*, 2014.

[15] S. Maheshwary and A. Kaul, “A hybrid attention network for fake news detection,”

[16] S. Maheshwary and H. Misra, “Matching resumes to jobs via deep siamese network,” in *Companion Proceedings of the The Web Conference 2018*, pp. 87–88, 2018.

[17] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, “Squad: 100,000+ questions for machine comprehension of text,” *arXiv preprint arXiv:1606.05250*, 2016.

[18] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh, “Vqa: Visual question answering,” in *Proceedings of the IEEE international conference on computer vision*, pp. 2425–2433, 2015.

[19] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.

[20] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, “Language models are few-shot learners,” *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[21] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” *Advances in neural information processing systems*, vol. 27, 2014.

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in neural information processing systems*, pp. 5998–6008, 2017.

[23] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” *Advances in neural information processing systems*, vol. 28, 2015.

[24] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, “End-to-end object detection with transformers,” in *European conference on computer vision*, pp. 213–229, Springer, 2020.

[25] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[26] C.-C. Chiu, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, E. Gonina, *et al.*, "State-of-the-art speech recognition with sequence-to-sequence models," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4774–4778, IEEE, 2018.

[27] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[28] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695, 2022.

[29] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE access*, vol. 8, pp. 58443–58469, 2020.

[30] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[31] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[32] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *2016 IEEE European symposium on security and privacy (EuroS&P)*, pp. 372–387, IEEE, 2016.

[33] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 ieee symposium on security and privacy (sp)*, pp. 39–57, Ieee, 2017.

[34] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2574–2582, 2016.

[35] N. Carlini and D. Wagner, "Audio adversarial examples: Targeted attacks on speech-to-text," in *2018 IEEE Security and Privacy Workshops (SPW)*, pp. 1–7, IEEE, 2018.

[36] J. Ebrahimi, A. Rao, D. Lowd, and D. Dou, "Hotflip: White-box adversarial examples for text classification," *arXiv preprint arXiv:1712.06751*, 2017.

[37] J. Ebrahimi, D. Lowd, and D. Dou, "On adversarial examples for character-level neural machine translation," *arXiv preprint arXiv:1806.09030*, 2018.

[38] J. Gao, J. Lanchantin, M. L. Soffa, and Y. Qi, "Black-box generation of adversarial text sequences to evade deep learning classifiers," in *2018 IEEE Security and Privacy Workshops (SPW)*, pp. 50–56, IEEE, 2018.

[39] B. Liang, H. Li, M. Su, P. Bian, X. Li, and W. Shi, "Deep text classification can be fooled," *arXiv preprint arXiv:1704.08006*, 2017.

[40] M. Alzantot, Y. Sharma, A. Elgohary, B.-J. Ho, M. Srivastava, and K.-W. Chang, "Generating natural language adversarial examples," *arXiv preprint arXiv:1804.07998*, 2018.

[41] J. Li, S. Ji, T. Du, B. Li, and T. Wang, "Textbugger: Generating adversarial text against real-world applications," *arXiv preprint arXiv:1812.05271*, 2018.

[42] S. Ren, Y. Deng, K. He, and W. Che, "Generating natural language adversarial examples through probability weighted word saliency," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 1085–1097, 2019.

[43] D. Jin, Z. Jin, J. T. Zhou, and P. Szolovits, "Is bert really robust? natural language attack on text classification and entailment," *arXiv preprint arXiv:1907.11932*, 2019.

[44] Y. Zang, F. Qi, C. Yang, Z. Liu, M. Zhang, Q. Liu, and M. Sun, "Word-level textual adversarial attacking as combinatorial optimization," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 6066–6080, 2020.

[45] S. Garg and G. Ramakrishnan, "Bae: Bert-based adversarial examples for text classification," *arXiv preprint arXiv:2004.01970*, 2020.

[46] L. Li, R. Ma, Q. Guo, X. Xue, and X. Qiu, "Bert-attack: Adversarial attack against bert using bert," *arXiv preprint arXiv:2004.09984*, 2020.

[47] Z. Zhao, D. Dua, and S. Singh, "Generating natural adversarial examples," *arXiv preprint arXiv:1710.11342*, 2017.

[48] P. Vijayaraghavan and D. Roy, "Generating black-box adversarial examples for text classifiers using a deep reinforced model," *arXiv preprint arXiv:1909.07873*, 2019.

[49] G. A. Miller, "Wordnet: a lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[50] N. Mrkšić, D. O. Séaghdha, B. Thomson, M. Gašić, L. Rojas-Barahona, P.-H. Su, D. Vandyke, T.-H. Wen, and S. Young, "Counter-fitting word vectors to linguistic constraints," *arXiv preprint arXiv:1603.00892*, 2016.

[51] J. Y. Yoo, J. X. Morris, E. Lifland, and Y. Qi, "Searching for a search method: Benchmarking search algorithms for generating nlp adversarial examples," *arXiv preprint arXiv:2009.06368*, 2020.

[52] D. Jin, Z. Jin, J. T. Zhou, and P. Szolovits, "Is bert really robust? a strong baseline for natural language attack on text classification and entailment," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, pp. 8018–8025, 2020.

[53] W. Brendel, J. Rauber, and M. Bethge, "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models," in *International Conference on Learning Representations*, 2018.

[54] M. Cheng, T. Le, P.-Y. Chen, J. Yi, H. Zhang, and C.-J. Hsieh, "Query-efficient hard-label black-box attack: An optimization-based approach," *arXiv preprint arXiv:1807.04457*, 2018.

[55] R. Jia and P. Liang, "Adversarial examples for evaluating reading comprehension systems," *arXiv preprint arXiv:1707.07328*, 2017.

[56] E. Wallace, S. Feng, N. Kandpal, M. Gardner, and S. Singh, "Universal adversarial triggers for attacking and analyzing nlp," *arXiv preprint arXiv:1908.07125*, 2019.

[57] B. Wang, H. Pei, B. Pan, Q. Chen, S. Wang, and B. Li, "T3: Tree-autoencoder constrained adversarial text generation for targeted attack," *arXiv preprint arXiv:1912.10375*, 2019.

[58] Y. Cheng, L. Jiang, and W. Macherey, "Robust neural machine translation with doubly adversarial inputs," *arXiv preprint arXiv:1906.02443*, 2019.

[59] X. Zhang, J. Zhang, Z. Chen, and K. He, "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)*, pp. 1967–1977, 2021.

[60] E. Dinan, S. Humeau, B. Chintagunta, and J. Weston, "Build it break it fix it for dialogue safety: Robustness from adversarial human attack," *arXiv preprint arXiv:1908.06083*, 2019.

[61] R. Maheshwary, S. Maheshwary, and V. Pudi, "Generating natural language attacks in a hard label black box setting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 13525–13533, 2021.

[62] R. Maheshwary, S. Maheshwary, and V. Pudi, "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*, (Online and Punta Cana, Dominican Republic), pp. 8396–8409, Association for Computational Linguistics, Nov. 2021.

[63] M. Glockner, V. Shwartz, and Y. Goldberg, "Breaking nli systems with sentences that require simple lexical inferences," *arXiv preprint arXiv:1805.02266*, 2018.

[64] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pp. 506–519, 2017.

[65] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *2016 IEEE symposium on security and privacy (SP)*, pp. 582–597, IEEE, 2016.

[66] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin, "Black-box adversarial attacks with limited queries and information," *arXiv preprint arXiv:1804.08598*, 2018.

[67] L. Song, X. Yu, H.-T. Peng, and K. Narasimhan, "Universal adversarial attacks with natural triggers for text classification," *arXiv preprint arXiv:2005.00174*, 2020.

[68] J. Zhao, Y. Kim, K. Zhang, A. Rush, and Y. LeCun, "Adversarially regularized autoencoders," in *International conference on machine learning*, pp. 5902–5911, PMLR, 2018.

[69] M. Sato, J. Suzuki, H. Shindo, and Y. Matsumoto, "Interpretable adversarial perturbation in input embedding space for text," *arXiv preprint arXiv:1805.02917*, 2018.

[70] C. Guo, A. Sablayrolles, H. Jégou, and D. Kiela, "Gradient-based adversarial attacks against text transformers," *arXiv preprint arXiv:2104.13733*, 2021.

[71] H. Zhang, H. Zhou, N. Miao, and L. Li, "Generating fluent adversarial examples for natural languages," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 5564–5569, 2019.

[72] D. Li, Y. Zhang, H. Peng, L. Chen, C. Brockett, M.-T. Sun, and B. Dolan, "Contextualized perturbation for textual adversarial attack," *arXiv preprint arXiv:2009.07502*, 2020.

[73] T. Wang, X. Wang, Y. Qin, B. Packer, K. Li, J. Chen, A. Beutel, and E. Chi, "Cat-gen: Improving robustness in nlp models via controlled adversarial text generation," *arXiv preprint arXiv:2010.02338*, 2020.

[74] S. Welleck, K. Brantley, H. D. Iii, and K. Cho, "Non-monotonic sequential text generation," in *International Conference on Machine Learning*, pp. 6716–6726, PMLR, 2019.

[75] Z. Dong and Q. Dong, *Hownet and the computation of meaning (with Cd-rom)*. World Scientific, 2006.

[76] E. Wallace, M. Stern, and D. Song, "Imitation attacks and defenses for black-box machine translation systems," *arXiv preprint arXiv:2004.15015*, 2020.

[77] X. He, L. Lyu, Q. Xu, and L. Sun, "Model extraction and adversarial transferability, your bert is vulnerable!," *arXiv preprint arXiv:2103.10013*, 2021.

[78] L. Yuan, X. Zheng, Y. Zhou, C.-J. Hsieh, and K.-W. Chang, "On the transferability of adversarial attacksagainst neural text classifier," *arXiv preprint arXiv:2011.08558*, 2020.

[79] W. C. Gan and H. T. Ng, "Improving the robustness of question answering systems to question paraphrasing," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 6065–6075, 2019.

[80] M. Iyyer, J. Wieting, K. Gimpel, and L. Zettlemoyer, "Adversarial example generation with syntactically controlled paraphrase networks," 2018.

[81] M. T. Ribeiro, S. Singh, and C. Guestrin, "Semantically equivalent adversarial rules for debugging nlp models," in *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2018.

[82] Y. Gil, Y. Chai, O. Gorodissky, and J. Berant, "White-to-black: Efficient distillation of black-box adversarial attacks," *arXiv preprint arXiv:1904.02405*, 2019.

[83] W. Zou, S. Huang, J. Xie, X. Dai, and J. Chen, "A reinforced generation of adversarial examples for neural machine translation," *arXiv preprint arXiv:1911.03677*, 2019.

[84] Y. Wang, X. Liu, and S. Shi, "Deep neural solver for math word problems," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, (Copenhagen, Denmark), pp. 845–854, Association for Computational Linguistics, Sept. 2017.

[85] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[86] T.-R. Chiang and Y.-N. Chen, "Semantically-aligned equation generation for solving and reasoning math word problems," *arXiv preprint arXiv:1811.00720*, 2018.

[87] L. Wang, D. Zhang, J. Zhang, X. Xu, L. Gao, B. Dai, and H. Shen, "Template-based math word problem solvers with recursive neural networks," in *AAAI*, 2019.

[88] Z. Xie and S. Sun, "A goal-driven tree-structured neural model for math word problems.," in *IJCAI*, pp. 5299–5305, 2019.

[89] J. Zhang, L. Wang, R. K.-W. Lee, Y. Bin, Y. Wang, J. Shao, and E.-P. Lim, "Graph-to-tree learning for solving math word problems," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, (Online), pp. 3928–3937, Association for Computational Linguistics, July 2020.

[90] R. Koncel-Kedziorski, S. Roy, A. Amini, N. Kushman, and H. Hajishirzi, “MAWPS: A math word problem repository,” in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, (San Diego, California), pp. 1152–1157, Association for Computational Linguistics, June 2016.

[91] S.-y. Miao, C.-C. Liang, and K.-Y. Su, “A diverse corpus for evaluating and developing English math word problem solvers,” in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, (Online), pp. 975–984, Association for Computational Linguistics, July 2020.

[92] A. Amini, S. Gabriel, P. Lin, R. Koncel-Kedziorski, Y. Choi, and H. Hajishirzi, “Mathqa: Towards interpretable math word problem solving with operation-based formalisms,” 2019.

[93] D. Huang, S. Shi, C.-Y. Lin, J. Yin, and W.-Y. Ma, “How well do computers solve math word problems? large-scale dataset construction and evaluation,” in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (Berlin, Germany), pp. 887–896, Association for Computational Linguistics, Aug. 2016.

[94] X. Zhang, J. Zhao, and Y. LeCun, “Character-level convolutional networks for text classification,” in *Advances in neural information processing systems*, pp. 649–657, 2015.

[95] B. Pang and L. Lee, “Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales,” in *Proceedings of the 43rd annual meeting on association for computational linguistics*, pp. 115–124, Association for Computational Linguistics, 2005.

[96] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, “Learning word vectors for sentiment analysis,” in *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pp. 142–150, Association for Computational Linguistics, 2011.

[97] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning, “A large annotated corpus for learning natural language inference,” *arXiv preprint arXiv:1508.05326*, 2015.

[98] A. Williams, N. Nangia, and S. R. Bowman, “A broad-coverage challenge corpus for sentence understanding through inference,” *arXiv preprint arXiv:1704.05426*, 2017.

[99] Q. Chen, X. Zhu, Z. Ling, S. Wei, H. Jiang, and D. Inkpen, “Enhanced lstm for natural language inference,” *arXiv preprint arXiv:1609.06038*, 2016.

[100] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes, “Supervised learning of universal sentence representations from natural language inference data,” *arXiv preprint arXiv:1705.02364*, 2017.

[101] D. Cer, Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, *et al.*, "Universal sentence encoder," *arXiv preprint arXiv:1803.11175*, 2018.

[102] Z. Dong and Q. Dong, "Hownet-a hybrid language and knowledge resource," in *International Conference on Natural Language Processing and Knowledge Engineering, 2003. Proceedings. 2003*, pp. 820–824, IEEE, 2003.

[103] T. Niven and H.-Y. Kao, "Probing neural network comprehension of natural language arguments," *arXiv preprint arXiv:1907.07355*, 2019.

[104] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pp. 1480–1489, 2016.

[105] A. P. Parikh, O. Täckström, D. Das, and J. Uszkoreit, "A decomposable attention model for natural language inference," *arXiv preprint arXiv:1606.01933*, 2016.

[106] M. S. Charikar, "Similarity estimation techniques from rounding algorithms," in *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pp. 380–388, 2002.

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

[108] R. Maheshwary, S. Maheshwary, and V. Pudi, "A context aware approach for generating natural language attacks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 15839–15840, 2021.

[109] R. Koncel-Kedziorski, S. Roy, A. Amini, N. Kushman, and H. Hajishirzi, "Mawps: A math word problem repository," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1152–1157, 2016.

[110] S.-Y. Miao, C.-C. Liang, and K.-Y. Su, "A diverse corpus for evaluating and developing english math word problem solvers," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 975–984, 2020.

[111] D. Huang, S. Shi, C.-Y. Lin, J. Yin, and W.-Y. Ma, "How well do computers solve math word problems? large-scale dataset construction and evaluation," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 887–896, 2016.

[112] D. Saxton, E. Grefenstette, F. Hill, and P. Kohli, "Analysing mathematical reasoning abilities of neural models," *arXiv preprint arXiv:1904.01557*, 2019.

[113] Y. Wang, X. Liu, and S. Shi, "Deep neural solver for math word problems," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 845–854, 2017.

[114] J. Zhang, L. Wang, R. K.-W. Lee, Y. Bin, Y. Wang, J. Shao, and E.-P. Lim, "Graph-to-tree learning for solving math word problems," Association for Computational Linguistics, 2020.

[115] A. Patel, S. Bhattamishra, and N. Goyal, "Are NLP models really able to solve simple math word problems?," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, (Online), pp. 2080–2094, Association for Computational Linguistics, June 2021.

[116] R. T. McCoy, E. Pavlick, and T. Linzen, "Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference," *arXiv preprint arXiv:1902.01007*, 2019.

[117] V. Kumar, R. Maheshwary, and V. Pudi, "Adversarial examples for evaluating math word problem solvers," in *Findings of the Association for Computational Linguistics: EMNLP 2021*, (Punta Cana, Dominican Republic), pp. 2705–2712, Association for Computational Linguistics, Nov. 2021.

[118] J. Mallinson, R. Sennrich, and M. Lapata, "Paraphrasing revisited with neural machine translation," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, (Valencia, Spain), Association for Computational Linguistics, Apr. 2017.

[119] J. Zhang, Y. Zhao, M. Saleh, and P. Liu, "Pegasus: Pre-training with extracted gap-sentences for abstractive summarization," in *International Conference on Machine Learning*, pp. 11328–11339, PMLR, 2020.

[120] V. Kumar, R. Maheshwary, and V. Pudi, "Practice makes a solver perfect: Data augmentation for math word problem solvers," in *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, (Seattle, United States), pp. 4194–4206, Association for Computational Linguistics, July 2022.