Skoltech

Skolkovo Institute of Science and Technology

MASTER'S THESIS

Adversarial Attacks on Symbolic Sequence Classifiers

Master's Educational Program: Data Science

Student Ivan Fursov

Research Advisor: Evgeny Burnaev, Professor

Co-Advisor: Alexey Zaytsev, Research Scientist

Moscow 2020

Skoltech

Skolkovo Institute of Science and Technology

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Адверсальные Атаки на Классификаторы Символьных Последовательностей

Магистерская образовательная программа: Науки о данных

Студент:	Фурсов Иван Андреевич
Научный руководитель:	Бурнаев Евгений Владимирович Доцент
Со-руководитель:	Зайцев Алексей Алексеевич Научный сотрудник

Москва, 2020

Adversarial Attacks on Symbolic Sequence Classifiers

by

Ivan Fursov

Friday 29th May, 2020 23:15

Submitted to the Skoltech Center of Research, Entrepreneurship and Innovation
on June 2020, in partial fulfillment of the requirements for the
MSc, Data Science

Abstract

An adversarial attack paradigm explores various scenarios for the vulnerability of machine and especially deep learning models: minor changes of the input can force a model failure. Most of the state of the art frameworks focus on adversarial attacks for images and other structured model inputs. The adversarial attacks for categorical sequences can also be harmful if they are successful.

However, successful attacks for inputs based on categorical sequences are challenging because of the non-differentiability of the target function with respect to the input. We handle these challenges using a loss-based attack. We use a state-of-the-art LM model for adversarial attacks either as a generator of adversarial examples or as a general language model. To achieve high performance, we use direct optimization of edit distance and classifier score via smooth approximation.

As a result, we obtain semantically better samples. Moreover, they are resistant to adversarial training and adversarial detectors. These two properties are hard to achieve for state-of-the-art adversarial examples generators that also often take into account the data nature. On the contrary, our model works for diverse datasets on money transactions, medical fraud, and NLP datasets.

Dedicated to my parents.

Acknowledgments

I would like to thank my research advisor Professor Evgeny Burnaev for his valuable suggestions and ideas.

I would like to express my gratitude to my research co-advisor Alexey Zaytsev for his patience and constant support in solving any problem I faced.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The deep learning revolution has led to the usage of deep neural network-based models across all sectors in the industry: from self-driving cars Sejnowski [2018] to oil and gas Sudakov et al. [2019]. However, the reliability of these solutions are questionable due to the vulnerability of almost all of the deep learning models to adversarial attacks Yuan et al. [2019] including models in computer vision Akhtar and Mian [2018], Khrulkov and Oseledets [2018], NLP Zhang et al. [2019], Wang et al. [2019], and graphs Sun et al. [2018]. The idea of an adversarial attack is to generate an object that fools a deep learning model by inserting changes into the original object, undetectable to a human eye: a deep learning model misclassifies the generated object. In contrast, for a human, it is evident that the class of the object remains the same Kurakin et al. [2017].

For images, we can calculate derivatives of the class probabilities with respect to the color of pixels. Thus, moving in the direction opposite to the gradient, we can apply slight alterations to a few pixels and get a misclassified image while keeping the image almost the same. For different problem, statements attacks can be different, but in general, a continuous space of images is rich enough for providing images that fool deep learning models.

The situation is different for sequential data, due to its discrete categorical nature. Whilst an object representation can lie in a continuous space, when we go back to the space of sequences, we can move far away from the initial object, rendering partial derivatives useless. The space of possible modification is also limited. For certain problems a malicious user can not modify an object arbitrarily. For example, whilst trying to increase a credit score we can not remove a transaction from the history of transactions available to the bank; we can only add another transaction. Both of these challenges impose additional constraints for creation of adversarial attacks for categorical sequential data.

A survey on adversarial attacks for sequences Zhang et al. [2019], Wang et al. [2019] presents a list of possible options to overcome these difficulties. With respect to white-box attacks, there are two main research directions at the moment. Many approaches that accept the initial space of tokens as input attempt to modify these sequences using operations like addition, replacement, or switching of tokens Samanta and Mehta [2017], Liang et al. [2017], Ebrahimi et al. [2018b]. Another idea is to move into an embedded space and leverage on gradients and optimization approaches in that space Sato et al. [2018]. We also note that most of these works focus on text sequence data.

We propose two approaches that can alleviate the problems mentioned above with differentiability and limited space of modification actions, and work mostly in the space of embedded sequences. The first approach is a new strong baseline that samples sequences from a pre-trained Language Model. By varying the temperature and the number of samples, we can generate diverse and semantically correct adversarial examples. The second approach illustrates that we can adopt differentiable versions of sequential distance metrics. We use a trained differentiable version of the Levenshtein distance Moon et al. [2018b]. In this case, our loss is differentiable, and we can adopt any gradient-based adversarial attack. The two approaches, which we name SampleFool and CASCADA attacks, are summarised explained in chapter 3.

As a generative model for adversarial attacks, we use a masked LM model Devlin et al. [2018], thus allowing the constructed model to be reused for generating adversarial attacks based on these two approaches and create adversarial attacks with a target direction. The validation of our approaches includes testing on diverse datasets from NLP, bank transactions, and medical insurance domains.

To sum up, the main contributions of this work are the following.

- We consider the problem of adversarial attack generation for categorical sequential data.

- Our first approach is based on an adaptation of MLM for sampling adversaries.
- Our second approach uses continuous relaxation of the initial problem. This makes it possible to perform a classic gradient-based adversarial attack after applying a few new tricks.
- We construct LM models to generate adversarial attacks via Gumbel Estimator and test the performance for attacking models based on standard classifiers, e.g., CNN-based and GRU-based architectures.
- Our adversarial attacks are resistant to defenses: adversarial training and adversaries detectors and demonstrate comparable performance for defenseless models.

The deep learning revolution has led to the extensive usage of deep learning usage for many tasks. Such models are being used for all sorts of data: tables, images, videos, graphs, texts, sequences, and many more. However, these models are far away from perfection and human-level understanding for most of the tasks. Models are prompt to error, especially for adversarial examples. Szegedy et al. [2014] Biggio et al. [2013]

Adversarial examples are a special form of the data designed to cause the model to output undesired answers. For classification tasks, it is a misclassification; for segmentation, it is an inability to segment objects, for reading comprehension tasks, it is a wrong answer, and so on. Adversarial attacks expose vulnerabilities of machine learning models and can be used in the wrong hands to bypass essential systems. A defaulter can receive a high credit score and get a loan by faking his transactions. Patients can receive unnecessary treatments, and insurance companies lose millions of dollars. Particular phrases may cause chatbot systems to output racial statements, and the chatbot systems' owners will have to shut down the system and carry lawsuits. Road signs can be physically modified in the way the autonomous car systems fail to recognize the sign and can lead to severe consequences Chen et al. [2019]. Security concerns grow as the number of vital systems is deployed into production, and the human component is replaced by machines.

These carefully curated examples are correctly classified by a human observer but can fool a target model, raising serious concerns regarding the security and integrity of existing ML algorithms.

Adversarial attacks can also be used for model evaluation and interpretation. Adversarial attacks may find limitations of models and identify the direction of future improvements of models. Adversaries find weak spots in the model: it may be the signs of overfitting/underfitting; it also may be an inability to generalize for some types of data.

While existing adversarial generation algorithms achieved success in image and speech domains, it is a challenging task to deal with sequence data (e.g., natural language sentences) due to its discrete and non-differential nature.

1.1 Contributions

This work has the following contributions in adversarial attack generation for NLP and non-NLP data.

- Our proposed new adversarial attacks generation algorithm CASCADA is capable of generating coherent, grammar, and syntax-correct adversarial examples in an end-to-end manner.
- We introduce a direct optimization of edit-distance between original and adversarial sequences using a smooth approximation Deep Levenshtein
- We propose a sophisticated metric to evaluate the success of an attack.
- We demonstrate that the proposed algorithm is resistant to conventional defense techniques: adversarial training and adversarial detection.

1.2 Thesis Structure

Chapter 2 - Background In this section, we introduce the problem and review the literature and related work.

Chapter 3 - Methods In this section, we discuss the novel algorithm we propose.

Chapter 4 - Experiments In this chapter, we discuss our results obtained on the sequence classification task. Examine the robustness of defense techniques.

Chapter 5 - Conclusion In this chapter, we discuss our results obtained on the sequence classification task. We conduct numerous experiments on NLP and non-NLP datasets and compare our algorithms with existing approaches. We also examine the robustness of our algorithms to defense techniques.

"Training even bigger ConvNets and LSTMs on ever bigger datasets gets us closer to Strong AI — in the same sense that building taller towers gets us closer to the moon."

François Chollet

Chapter 2

Background

2.1 Definitions and notations

In this section, we provide the definitions of adversarial attacks and introduce basic algorithms and defense techniques. We explain the difference between perturbation measurement in sequential data (NLP) and non-sequential (CV). We give an intuition of the attacks evaluation and the current status in academia. This section briefly explains the critical components of adversarial attacks and countermeasures.

2.1.1 Definitions

- **Deep Neural Network** (DNN). A DNN is a non-linear differentiable function $f_\theta : X \longrightarrow Y$ that maps features X into a prediction Y which can be positive integers representing classes, a real number representing a regression problem or any other type depending on the problem. θ is a set of randomly initialized and trained using a loss function $L(f_\theta(X), Y)$ which minimizes the difference between predictions and observed data.

- **Adversarial example**. An adversarial example x'_j is an input x_j modified by an adversarial generation examples algorithm. An adversarial example $\hat{x}_j$ is obtained from the problem of finding an input such that it has different compared to a ground truth label $f_\theta(\hat{x}_j) \neq y_j$.

- **Pertrubations**. In order to change the model's prediction, perturbations should be introduced to the initial sequence. For data such as images, adversaries can add noise ϵ to the original image.

$$x'_j = x_j + \epsilon$$
$$f_\theta(x'_j) \neq y_j$$

However, perturbations for sequential data are of a different type. They can be applied by modifying the sequence via deletion/substitution/insertion of tokens.

2.1.2 Models

- **Threat Model**. As defined in Yuan et al. [2017], the Threat Model is a scenario where the adversaries can attack only at the deploying stage after the victim model is trained. The adversaries may have the knowledge of victim models, but the knowledge is limited, depending on the case. In this setting, the adversaries are not allowed to modify the model's parameters or change the models' architecture.
- **Victim models**. A victim model is a model the adversaries attack for personal gain. The victim model can be a conventional ML model but also a DL model. In this work, we investigate the DL models.

2.1.3 Attacks

Many new attack types appear in academia. Types of adversaries depend on assumptions about what is available for the adversary. Here we discuss common adversary's goals and knowledge.

Adversary's Goal

For Threat Model settings, the adversary's goal is either target or non-targeted. Given (x, y) pair, the adversary changes x in the way that the ground truth label y

changes. For targeted attacks, the adversarial algorithm changes the prediction to a specific prediction $y^{'}$, while non-targeted attacks are asked to change the model's prediction to another. The latter type is an easier one since the adversary has more room to maneuver.

Adversary's Knowledge

There are two main strategies in terms of the knowledge available for the adversary: black-box and white-box. In the first case, the adversary does not have access to the model's architecture, parameters, loss function. It is only able to obtain the model's predictions by querying it. On the other hand, a white-box attack is a type of attack when the adversarial algorithm has access to all the model's information. Usually, white-box attacks are used for model interpretation and more robust training, while black-box attacks are for the degree of protection testing.

2.1.4 Evaluation

An attack is successful if the adversarial example is similar to the original input, and the model output an incorrect prediction.

Adversarial attacks should be designed in a way when human eyes do not perceive them, therefore, humans can correctly determine what the model's output should be. In practice, it is hard to tell whether the original input has changed for the human eye because it should be labeled by humans, which is expensive and lengthy.

To approximate the human eye perception criterion, researchers use the Perturbation constraint. For computer vision problems, x' should not change the right output, but it should be able to fool the model. Therefore, the noise added to the original image x must not be small so that the output changes and must not be significant so that the human does not change his mind about it. Usually, researchers use thresholds to bound the L_p norms. Goodfellow et al. [2014] use max-norms $\|\eta\|_\infty \leq \epsilon$. However, other norms L_0, L_2 are also a common choice.

For sequential data, discrete operations such as insertions and deletions add noise. Thus L_p norms can not be used for perturbation constraint measure. To evaluate perturbations, one can use syntax and semantic distance evaluations.

A common way to calculate the syntax changes in sequences is edit-distance (Levenshtein distance) Cormode and Muthukrishnan [2007]. The edit distance between two sequences is defined to be the minimum number of character inserts, deletes, and changes needed to convert one sequence into the other.

The semantic meaning of a sequence may change only by changing one token. For example, in NLP, one can replace a word in a sentence with another word, and the sentence completely loses its meaning or changes it. One simple and effective way to measure the grammar component in a sequence is to calculate the perplexity of a sequence. Minervini and Riedel [2018] use perplexity to validate the adversarial sequences are valid.

We discuss the choice of metrics in chapter 4.

2.1.5 Countermeasures

There are many ways to defend deep learning models from the threat of adversaries. The work Xu et al. [2019] proposes to use the following categorization of countermeasures.

1. Gradient Masking/Obfuscation Since most attack algorithms are based on the gradient information of the classifier, masking or hiding the gradients will confound the adversaries

2. Robust Optimization. Re-learning a DNN classifier's parameters can increase its robustness. The trained classifier will correctly classify the subsequently generated adversarial examples.

3. Adversarial Example Detection. Adversaries detection is a binary classification of natural vs. adversarial examples. The algorithm detects adversarial examples and disallows input into the classifier; instead, it sends the adversarial query to the human who distinguishes adversaries.

According to Yuan et al. [2017], there are two types of defense strategies:

1. Reactive: detect adversarial examples after deep neural networks are built.

2. Proactive: make deep neural networks more robust before adversaries generate adversarial examples.

In order to build a robust system, both types of defenses should be taken into account. However, we should mention, that defense from adversaries is an iterative process: attackers invent more sophisticated ways to trick models, while developers try to maintain the systems and be resistant to attacks.

Almost all defenses are shown to be helpful only for a portion of adversarial attacks. They tend not to be defensive for some strong, unexpected, and unseen attacks. Most defenses target adversarial examples in the computer vision task. However, with the development of adversarial examples in other areas, new defenses for these areas, especially for safety-critical systems are urgently required.

2.2 Related Work

There exist adversarial attacks for different types of data. The most popular targets for adversarial attacks are images Szegedy et al. [2014], Goodfellow et al. [2014], although some work has also been done in areas such as graph data Zügner et al. [2018] and sequences Papernot et al. [2016b].

It seems that one of the first sources for the generation of adversarial attacks for discrete sequences is Papernot et al. [2016b]. The authors correctly identify the main challenges for adversarial attacks for discrete sequence models: a discrete space of possible objects and a elaborate definition of a semantically coherent sequence. Their approach focuses on a white-box adversarial attack with a binary classification problem. In our work, we also consider this problem statement but focus on black-box adversarial attacks for sequences and target vs. non-target attacks. This problem statement was considered in Gao et al. [2018], Liang et al. [2017], Jin et al. [2020]. Authors of Gao et al. [2018] identify specific pairs of tokens and then permute their positions within these pairs, thus working directly on a token level. Another black-box approach from Liang et al. [2017] also performs a direct search for the most harmful tokens, trying to fool a classifier.

The first research direction is a direct generation of sequences by trying to apply natural modifications like deletion or replacement of a token in a sequence, with the expectation that it will result in a sufficiently good adversarial outcome. Samanta and Mehta [2017]. This idea leads to an extensive search among the space of possible sequences, making the problem computationally challenging, especially if the inference time for a neural network is significant Fursov et al. [2019]. Moreover, we have little control over the proximity of the initial and modified sequences, as we do not take into account the proximity of tokens in the embedded space.

The second research direction is closer to the common approaches used by adversarial attacks practitioners, as we try to use gradients of the cost function and distance measure Sato et al. [2018]. However, the authors in Sato et al. [2018] limit directions of perturbations by moving towards another word in an embedded space, which seems restrictive for state-of-the-art RNNs that adopt mechanisms such as attention Vaswani et al. [2017] and beam search Wiseman and Rush [2016] to increase the performance of the sequence model.

To leverage the embeddings space for the generation of adversarial sequences, we need to define a differentiable mapping from the embedded space to the space of sequences. Another more straightforward solution would be to propose an effective procedure for traversing through the embedded space.

A differentiable loss function takes into account the distance between the initial and generated sequences and simultaneously tries to minimize the class score for the generated adversarial example. Authors of Cheng et al. [2018] compose a white-box adversarial attack based on this idea, but due to a high divergence between the embedded and categorical sequence spaces, they return to the categorical sequence space at each step, thus making the search ineffective and not end-to-end.

For the class score, it is relatively easy to define a differentiable version of it, while for the distance measure, this problem is more complicated. However, there are differentiable versions of distance measures, e.g., a differentiable BLEU score Zhukov et al. [2017], a learned Moon et al. [2018b] or soft Ofitserov et al. [2019] Levenshtein edit distance.

As we see from the current state of the art, there is a remaining need to identify an effective way to explore the space of categorical sequences for the problem of generation of adversarial attacks. Moreover, as most of the applications focus on NLP-related tasks, there is room for improvement by widening the scope of application domains for adversarial attacks on categorical sequences.

2.3 Thesis objectives

In this chapter, we define the goals and derive the specific questions to be addressed in our research.

We are developing an adversarial algorithm that is

- Universal – will work for any sequential data and any datasets.

  Figure 2-1 illustrates the differences between NLP and non-NLP data. In NLP data, one token is exactly one word (or a character), while for non-NLP tasks, one token may represent different states.

- Difficult to defend from using traditional defense strategies

- Produce "semantically"-correct adversarial examples

  In Figure 2-2 we see how the original sequence can be modified. In the first case, the sentence loses meaning, and such an adversary can be detected using spell-checkers for character-level attacks and syntax parsers for word-level attacks. The second attack changes the meaning of the sentence. However, it does not loses syntax and grammar structure of the sentence.

- Efficient– no need to re-train the algorithm for each dataset.

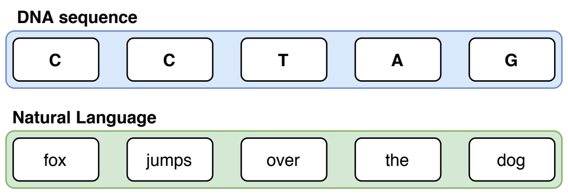


Figure 2-1: Different sequence types. Natural language sequences and DNA sequences.

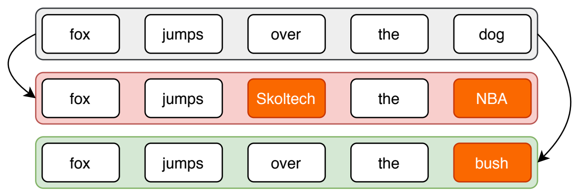


Figure 2-2: Semantically-correct and incorrect adversarial examples.

2.4 Tools

For all of our experiments, we used AllenNLP library Gardner et al. [2017], designed for developing state-of-the-art deep learning models on a wide variety of linguistic tasks. We adjust the library for non-NLP usage and train all the models treating the input as an arbitrary sequence. For additional tasks, we use PyTorch Paszke et al. [2019] deep learning library. All the experiments presented in this work were conducted on two **GTX 1080 Ti** GPUs with a 64 Gb RAM machine.

Chapter 3

Methods

In this section, we start with the description of components of the adversarial algorithm we propose, with some necessary details on model training and structure. We then describe the classifier model that we fool using our adversarial model. Next, we proceed with the description of how our model is used to generate adversarial examples. Finally, we describe how to obtain a differentiable version of the Levenshtein distance and adopt the Gumbel Estimator approach to our problem.

This section introduces CASCADA (**CA**tegorical **S**equences **C**ontinuous **AD**versarial **A**ttack) algorithm. CASCADA algorithm is a method to generate adversarial attacks on categorical sequences. It uses a masked language model pre-trained on a domain corpus, Deep Levenshtein model, which is a function that approximates edit distance between two sequences and a substitute model that is used for finding the needed adversarial version of the sequence.

The CASCADA algorithm can work both in black-box and white-box settings. In the first case, CASCADA uses a substitute model in which the algorithm attacks to obtain adversarial sequences and then transfers resulting sequences to the targeted model. In the second scenario, CASCADA has access to the targeted model, including access to gradients and outputs.

The CASCADA algorithm aims to modify the sequence in the way the attacked sequence and the adversarial sequence will be very similar to each other in terms of edit-distance, the resulting attack can occur in real life, meaning that this attack is hard to defend against.

While existing white-box and black-box approaches explore only the vulnerabilities of the current models they are attacking: find weak points and apply them to almost every attacked sequence. Exploring the model's vulnerabilities is an effective way to fool the model in the short run scenario. However, there are many approaches to defend against these attacks once and for all.

3.1 Masked LM

The masked language model (MLM) randomly masks some of the tokens from the input, and the objective is to predict the original vocabulary id of the masked sequence. The MLM can be seen as a generative model that captures the distribution of data. In Radford et al. [2018], the authors demonstrate that language models begin to learn question answering, machine translation, reading comprehension, and summarization tasks without any explicit supervision.

Language modeling is an unsupervised distribution task from a set of sequences $(x_1, x_2, \ldots, x_n)$ each of variable length $(l_1, l_2, \ldots, l_n)$. Tokens are assumed to be conditionally independent and the joint probability of a sequence factorizes into the product of conditional probabilities [Jelinek and Mercer, 1980, Bengio et al., 2003].

$$p(x) = \prod_{i=1}^{n} p\left(s_n | s_1, \ldots, s_{n-1}\right) \tag{3.1}$$

This assumption allows us to sample from the distribution and estimate the probability of an arbitrary sequence $p(x)$. Recently, self-attention architectures like the Transformer Vaswani et al. [2017] have made a huge impact on NLP, improving former SOTA methods.

Language Learning has been highly successful in Natural Language Processing. Usually, researchers train LM's on large text corpora and then fine-tune on downstream tasks (text classification, named-entity recognition, sentiment analysys, etc.) [Dai and Le, 2015, Peters et al., 2018, Radford, 2018, Devlin et al., 2018]. There are two common training objectives for language modeling: autoregressive (AR) language modeling and autoencoding (AE) Yang et al. [2019].

AR language modeling estimates the probability of a text corpus in an autoregressive manner [Dai and Le, 2015, Peters et al., 2018, Radford, 2018]. AR models factorizes the likelihood of a sequence x into a product:

$$p(\mathbf{x}) = \prod_{t=1}^{T} p\left(x_t | \mathbf{x}_{<t}\right)$$

Given a sequence $\mathbf{x} = [x_1, \cdots, x_T]$, AR language model maximizes the likelihood

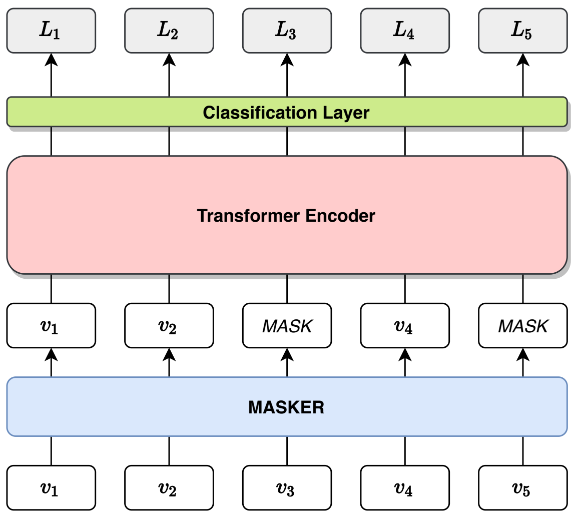


Figure 3-1: Masked Language Model architecture

under the forward autoregressive factorization:

$$\max_\theta \log p_\theta(\mathbf{x}) = \sum_{t=1}^{T} \log p_\theta\left(x_t | \mathbf{x}_{<t}\right) \tag{3.2}$$

On the other hand, AE learns to reconstruct the original sequence that was changed. The one the first models for this approach is BERT Devlin et al. [2018]. Bert masks random tokens in the initial sequences and learns to reconstruct them. For a sequence x, BERT first constructs a corrupted version $\hat{x}$ by randomly replacing tokens with a special symbol [MASK]. The training objective is to reconstruct $\bar{x}$ (masked tokens) from $\hat{x}$:

$$\max_\theta \quad \log p_\theta(\overline{\mathbf{x}} | \hat{\mathbf{x}}) \approx \sum_{t=1}^{T} m_t \log p_\theta\left(x_t | \hat{\mathbf{x}}\right) \tag{3.3}$$

, where $m_t = 1$ indicates x_t is masked.

In our experiments, we follow the ideas from Devlin et al. [2018] and train a BERT-like LM model. The architecture of the model is shown in Figure 3-1. We pass the sequence x. Masker module randomly adds noise to the sequence by masking and replacing some tokens $x \longrightarrow \hat{x}$. Then we pass the resulting sequence through the multi-layer bidirectional Transformer encoder Vaswani et al. [2017] and receive a hidden representation for each token in the sequence. We then use one fully-connected layer of the shape of the vocabulary size $|V|$ to reconstruct the sequence.

3.2 Deep Levenshtein

Levenshtein distance is a way of quantifying how dissimilar two sequences are to one another by counting the minimum number of operations required to transform one sequence into the other. Deep Levenshtein is a smooth approximation of a non-differentiable edit-distance function.

Initially, the idea to obtain an approximated version of edit distance comes from the string similarity search problem. There are two common tasks: similarity join and threshold search. Similarity join is a task of finding all sequence pairs $(s_i, s_j) \in \{s_1, s_2, \ldots, s_n\}$ with edit distance lower a certain threshold $\Delta_e(s_i, s_j) \leq t$ and $i < j$, where t is a threshold for the edit distance between pairs. Threshold search task finds all elements for a query sequence q that satisfy $\Delta_e(q, s) \leq t$. Both tasks are fundamental for similarity search applications and tried to be solved using numerous methods [Bayardo et al., 2007, Li et al., 2011, Zhang et al., 2010].

Edit-distance-based sequence similarity search has many applications in NLP, such as spell correction, data de-duplication, and sequence alignment Dai et al. [2020]. However, a brute-force approach is inefficient for large datasets, since the high computational complexity of edit distance is the main obstacle. For two sequences with length l, the complexity of calculating edit distance is equal to $O\left(l^2/\log(l)\right)$ Masek and Paterson [1980].

3.2.1 Metric Learning

Metric learning proved to be a reliable approach to solving fundamental tasks in computer vision: face recognition Liu et al. [2017], face verification Schroff et al. [2015], person re-identification Hermans et al. [2017]. Metric learning algorithms encode data to an embedding space, where similar objects are close together, and different ones are far apart.

A metric learning approach for edit distance can be defined by an embedding function $E(x)$ that maps sequences into fixed-length dense representations and a distance measure $d(E(x), E(y))$. The model is trained to approximate the distance in an original space $d(E(x), E(y)) \approx \Delta_e(x, y)$.

In Figure 3-2, the simplified version of the model's architecture is presented. Dai et al. [2020] use a convolutional neural network (CNN) as an encoder, while Zhang et al. [2020] train a GRU-based architecture and concatenate all the hidden states of a GRU to obtain a fixed-length representation of a sequence. Both approaches show superiority over other data-independent methods Chakraborty et al. [2016].

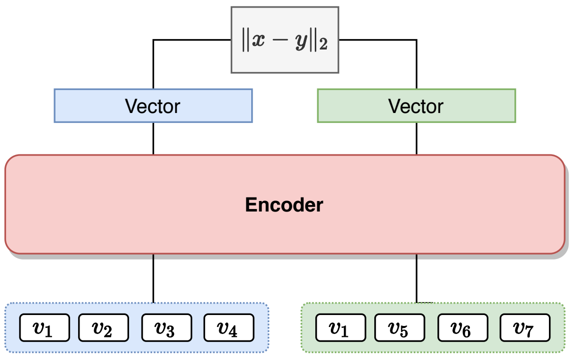


Figure 3-2: A metric-learning-based Deep Levenshtein model. We encode two variable-length sequences into fixed-length dense vectors x and y, then we calculate the euclidian distance between them.

We follow the notation of Moon et al. [2018a] and call a such model a Deep Levenshtein model. Moon et al. [2018a] propose to use Deep Levenshtein for lexical embeddings in the Multimodal Named Entity Disambiguation task. Authors apply bi-LSTM architecture and a cosine similarity score to approximate Levenshtein scores.

As a result, we have acquired pairs: each pair being a sequence and a close but different sequence obtained after the application of replacement. We have also added pairs obtained from the training data to get better coverage of far-away sequences. For each pair, we calculated $WER(x, y)$ and learned a shared-bidirectional CNN model $M(x_i)$ with the L2 loss objective $J(x, y) = (M(x) - M(y))^2$. To improve the quality of the model, we have used concatenations of $M(x)$ and corresponding attention vectors of $M(y)$.

Edit-distance is a function that measures the distance between two sequences. Since we work only with word-level scenarios, edit-distance transforms into Word Error Rate distance. We want to create adversarial attacks with low WER values since these attacks will be harder to capture and defend against.

3.2.2 Regression

In our experiments, we compare the metric learning approach with a deep learning regression model. We found that the regression approach improves the approximation error. However, this approach does not apply to the most string similarity search problems. Regression-based loses the opportunity to use the nearest neighbor search algorithms designed to accelerate the search in L_p norms Zhang et al. [2020].

For our adversarial algorithm described in section 3.4, we follow the ideas from Dai et al. [2020], where the authors use CNN-based architectures and metric-learning to train the model for edit-distance-based string similarity search. The authors emphasize that the data-driven approach works quite well for edit distance approximation.

We also found that the usage of embeddings for tokens representations works better than one-hot representations, as proposed in Dai et al. [2020]. The architecture is presented in Figure 3-3. The model receives two sequences (x, y), encodes it to a fixed-length dense representation using the shared encoder $z_x = E(x)$, $z_y = E(y)$. Then it concatenates the representations and the absolute difference between $\text{concat}(z_x, z_y, |z_x - z_y|)$ and uses one fully-connected layer with the $L2$ loss to approximate the Word Error Rate. We repeat after Dai et al. [2020] and use a CNN-based architecture.

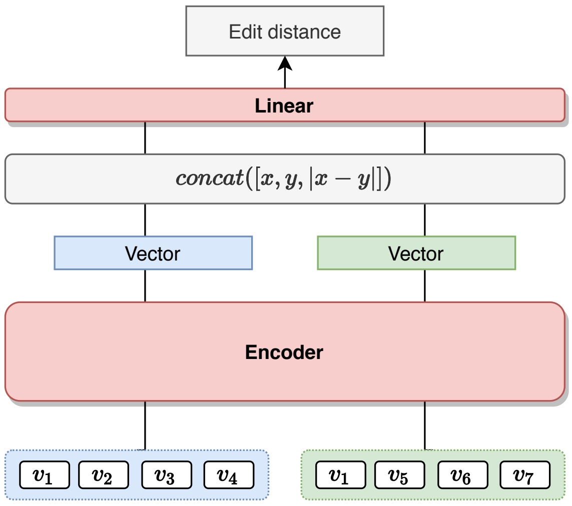


Figure 3-3: Deep Levenshtein, a regression version. We encode two variable-length sequences into fixed-length dense vectors x and y, then we concatenate x, y and $|x - y|$ and pass the result through one fully connected layer.

The quality of the model and the subsequent successful usage for adversarial attacks generation heavily depends on the dataset used for training. We discuss the dataset generation below.

We settled on the learning-based method since the approximation error is essential for the adversarial sequence generation. We abandoned the metric learning approach and focused on the regression with the L2 loss between the output of the architecture and the edit distance. The architecture is presented in Figure 3-3. The first part of the architecture is similar to the embedding learning approach. However, to reduce the approximation error, the model should explicitly know the differences between sequences. We add one layer above the concatenation of two vectors of the sequences and one vector of the absolute difference between these vectors.

3.2.3 Dataset Generation

To collect the training data, we have used examples generated from each dataset, around 2 million examples in total. For each example, we have applied sequence perturbations. We collected artificial examples using sequential changes to the original sequences. We randomly replaced one token with a random token from vocabulary for $\min(10, l_i)$ steps, where l_i is the length of x_i.

3.3 Gumbel-Softmax

Gumbel-Softmax trick Jang et al. [2016] is a technique that allows sampling to discrete random variables in a differentiable way. This trick is used in NLP for style transfer Shen et al. [2017], text modelling Yang et al. [2017], and text generation [Zhang et al., 2017, Fedus et al., 2018].

The continuous approximation to sampling from a categorical distribution is based on the Gumbel-Max Trick Maddison et al. [2014] and a concrete distribution Maddison et al. [2016].

The Gumbel-Max Trick Maddison et al. [2014] allows to sample from a categorical distribution $Cat(\alpha_1 \alpha_2, \ldots, \alpha_k)$, where the probability to observe a category k is α_k. This is trick is based on Gumbel Distribution:

$$CDF_{Gumbel}(\epsilon) = \exp(-\exp(-\epsilon)) \tag{3.4}$$

$$z = \text{one\_hot}\left(\arg\max_i [g_i + \log \pi_i]\right) \tag{3.5}$$

The work Jang et al. [2016] proves that $k_{\max} = \text{argmax}_{k' \leq K}\left(\epsilon_{k'} + \log\left(\alpha_{k'}\right)\right)$ follows the desired categorical distribution $Cat(\alpha_1 \alpha_2, \ldots, \alpha_k)$. To sample from this distribution, we generate an uniform random variable $u \in [0, 1]$ and use an inverse transform sampling procedure:

$$\begin{gathered} u \sim \mathcal{U}([0,1]) \\ \epsilon = CDF^{-1}_{\text{Gumbel}}(u) = -\log(-\log(u)) \end{gathered} \tag{3.6}$$

$$y_i = \frac{\exp\left(\left(\log\left(\pi_i\right) + g_i\right)/\tau\right)}{\sum_{j=1}^{k} \exp\left(\left(\log\left(\pi_j\right) + g_j\right)/\tau\right)} \quad \text{for } i = 1, \ldots, k \tag{3.7}$$

3.4 CASCADA

In this section, we discuss the way CASCADA works. CASCADA is a method for generating adversarial examples using a pre-trained language model and a deep Levenshtein model. CASCADA is a loss-based adversary that allows the algorithm to decide what tokens should be changed without introducing heuristics. It uses the Gumbel-Softmax estimator to sample from the categorical distribution so we can directly feed token indexes into the substitute model and a Deep Levenshtein.

We follow the ideas from controlled text generation task, Dathathri et al. [2019], where authors propose using a large pre-trained LM model. They leverage the power of GPT-2 model Radford [2018], which showed excellent performance for many NLP tasks and is a suitable encoder for text representations. Authors use attribute models $p(a|x)$ to control the generative model $p(x)$ to produce attribute-preserving texts. $p(a|x)$ is a differentiable classification model, attributes a are classes for which the text belongs to.

In our research, we pre-train a transformer encoder Vaswani et al. [2017] in a BERT Devlin et al. [2018] manner on a corresponding domain. We use all available data to train such a model.

Figure 3-4 illustrates the overall architecture of the algorithm. It consists of four main components: 1) pre-trained transformer Language Model $G(x) : x \longrightarrow p_\theta(x)$ 2) Straight-Through Gumbel Estimator $ST(p_\theta(x)) : p_\theta(x) \longrightarrow x'$ 3) Deep Levenshtein model $DL(x_i, x_j) : (x_i, x_j) \longrightarrow \mathbf{R}$ 3) A substitute classifier $C(x)$. We do not train this model to generate adversarial sequences. On the contrary, we use a pre-trained LM to generate coherent and semantically correct samples and modify LM's weights for each example to produce adversarial samples x'.

$G(x)$ aims to reconstruct the sequence. It encodes the original sequence x into the probability distribution of tokens $p_\theta(x)$. The length of the original sequence is L, and the shape of $p_\theta(x)$ is $L \times |V|$, where V is a vocabulary of the language. For non-NLP datasets, vocabulary is a set of all available events which may occur.

The Straight-Through Gumbel Estimator samples from the distribution of $p_\theta(x)$ and outputs the $|V|$ dimensional vectors z. For large values of τ, the expected value

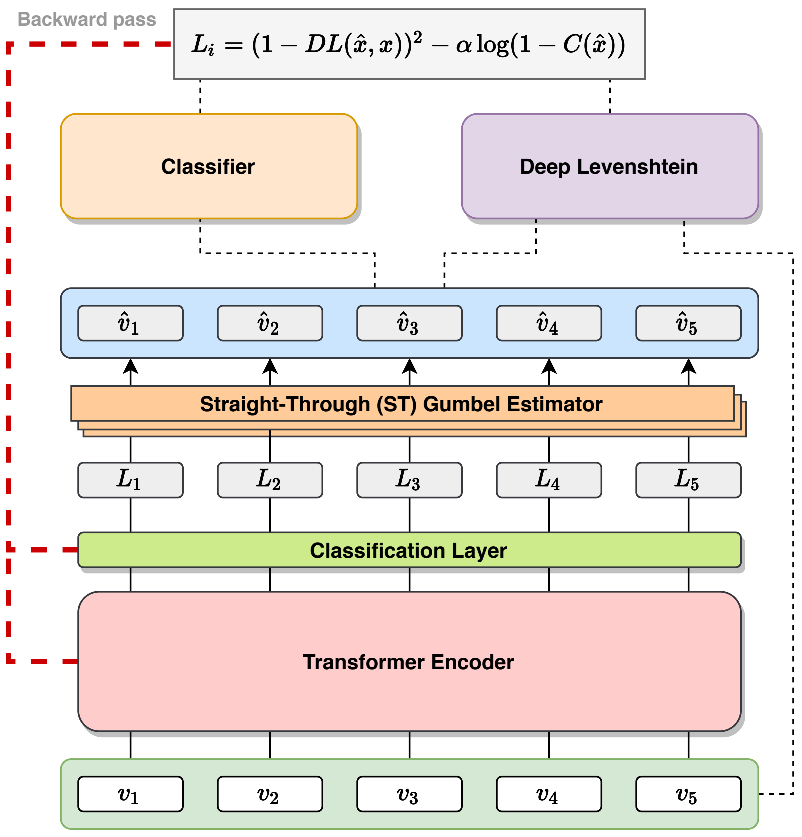

Backward pass
$L_i = (1 - DL(\hat{x}, x))^2 - \alpha \log(1 - C(\hat{x}))$
Classifier
Deep Levenshtein
$\hat{v}_1$ $\hat{v}_2$ $\hat{v}_3$ $\hat{v}_4$ $\hat{v}_5$
Straight-Through (ST) Gumbel Estimator
L_1 L_2 L_3 L_4 L_5
Classification Layer
Transformer Encoder
v_1 v_2 v_3 v_4 v_5

Figure 3-4: CASCADA architecture. **Step 1**: obtain logits from the pre-trained LM. **Step 2**: sample from logits using Gumbel-Softmax estimator. **Step 3**: Obtain probability $C(\hat{x}$ and edit distance $DL(\hat{x}, x)$. **Step 4**: Calculate loss, do a backward pass. **Step 5**: Update LM's weight using gradient descent

of a Gumbel-Softmax random variable converges to a uniform distribution, while for low values, the expected value approaches the expected value of a categorical distribution with the same logit. There is a tradeoff between temperatures that are close to zero $\tau \longrightarrow 0$, and samples are close to one-hot representation, but the variance of the gradients is large, and large temperatures $\tau \longrightarrow \infty$, where samples are smooth but the variance of the gradients is small. In practice, we find τ using Grid Search and sample multiple times to obtain better estimations. We found that $\tau = 1.5$ and 5 - 8 samples work fine for all datasets we investigate.

To pass the sequence to a classifier $C(x)$ and a Deep Levenshtein, we need to obtain one-hot representations of the sequence. However, the arg max operation cannot be applied, since it is not derivative. We follow the ideas from Jang et al. [2016] and discretize y using arg max but use our continuous approximation for estimating the gradient $\Delta x' \approx \Delta z$.

x'_i supposed to be adversarial, meaning that $C(x'_i) \neq y_i$. In order to be classified as intended by humans, we introduce the control of edit-distance between via Deep Levenshtein. Both an approximated edit-distance value $DL(x'_i, x_i)$ and a probability score $C(x_i)$ formed into a loss function.

$$J(x'_i, x_i, y_i) = (1 - DL(x'_i, x_i))^2 - \alpha \log(1 - C(x_i)[y_i]) \tag{3.8}$$

We penalize the model for any perturbations larger than 1 operation needed to transform from adversarial sequence to original. Since we focus on non-target attacks, the $C(x_i)[y_i]$ component is included in the loss. The smaller the probability of an attacked class, the smaller the loss.

Finally, we do a backward pass and update LM's weights. We found that updating the whole set of parameters θ is not the best strategy. Instead, using a grid search, we found out that updating the last linear layer and the last layer of the transformer works best.

3.4.1 Algorithm

```
Algorithm 1: CASCADA Attack Algorithm
  Input: Pre-trained $LM$, substitute classifier $C(x)$, Deep Levenshtein model
         $DL(x, y)$
  Data: Original sequence $x = [x_1, x_2, \dots, x_m]$ and true label
        $y \in [0, \dots, k-1]$
  Result: Adversarial sequence $x^* = A(x)$
  for $i \leftarrow 1$ to $N$ do
      $p_i := LM(x)$;
      $x^*_i := Gumbel(p_i)$ is an adversarial sequence;
      $C(x^*_i)[y]$ is a probability of class $y$;
      $DL(x^*_i, x)$ is an approximated WER;
      $L_i = (1 - DL(x^*_i, x))^2 - \alpha \log(1 - C(x^*_i)[y])$ ;
      $\theta_{LM} := \theta_{LM} - \beta \Delta L_i$ ;
  $x^* = \arg\max LM(x)$
```

The proposed approach for adversarial sequences generation is shown in algorithm 1, and consists of the following steps:

- **Step 1.** Pass the sequence x through the pre-trained LM. Obtain logits.

- **Step 2.** Sample an adversarial sequence from logits using Gumbel-Softmax Estimator.
- **Step 3.** Calculate probability $C(\hat{x}_i)$ and an approximated word error rate $DL(\hat{x}_i, x)$. Calculate loss value for the i-th iteration L_i
- **Step 4.** Do a backward pass. Update LM's weights using gradient descent.
- **Step 5.** obtain an adversarial sequence using argmax operation for the i-th iteration.

Note that the algorithm decides by itself which tokens should be replaced. The classification component in Equation 3.8 changes the sequence in a direction where the probability score $C(x')$ is low, and the Deel Levenshtein part does not allow the algorithm to completely change the sequence, keeping the adversarial sequence close to the original.

For each step of the model's optimization, we obtain an adversarial sequence, and for CASCADA with sampling, we get k adversarial samples each step. We do not take the last sequence as the final answer because the last steps are not always better than intermediate. It can be explained by the fact that the loss did not reach the plateau. To overcome this issue, we propose a simple selection strategy explained in algorithm 2.

Algorithm 2: Best adversary selection.

```
Input: A substitute classifier $C(x)$.
Data: A set of adversarial examples $[x'_1, x'_2, \ldots, x'_M]$. Original label $y$.
      Original sequence $x$.
Result: Adversarial sequence $x'$
changed_output = [...];
for $i \leftarrow 1$ to $M$ do
    if $x'_i \neq y$ then
        changed_output.append($x'_i$);
    else
        continue;
    end
end
if changed_output is not empty then
    $x' = \arg\min_j WER(x'_j, x)$;
else
    $x' = \arg\max_j (c(x) - c(x'_j))$;
end
```

3.4.2 Variations

Based on the ideas described above, we propose to use two basic variations of CASCADA. Moreover, we introduce a novel baseline for sequential adversarial attacks, which we call SamplingFool.

- **SamplingFool** works entirely on pre-trained Language Models. Since Language models learn a probability distribution over sequences, we can sample sequences from it. SamplingFool passes sequence x through the pretrained LM $p(.)$ and obtains logits z_i for each token in a vocabulary, then it converts logits into probabilities q_i by using a softmax with temperature T Hinton et al. [2015].

$$q_i = \frac{\exp\left(z_i/T\right)}{\sum_j \exp\left(z_j/T\right)} \tag{3.9}$$

Using $T > 1$ values produces a softer probability distribution over classes. As $T \longrightarrow \infty$, the original distribution approaches a uniform distribution.

Finally, SamplingFool samples sequences from categorical distribution with q_i probabilites. The sampled examples turn out to be good-looking and, as we discuss in chapter 4, can easily fool a classifier.

- **CASCADA with sampling**. In fact, we can replace argmax operation in algorithm 1 with a sampling from a categorical distribution procedure described above. In this setup, CASCADA will generate m adversarial examples for each iteration with almost no additional computational cost. We will see in chapter 4 that this approach works extremely well across all datasets.

Chapter 4

Experiments

In this section, we conduct experiments, propose a new metric, and discuss results. Just as in the computer vision, it is crucial for the adversarial attack generator not only change the output of the targeted model, but also, the adversarial object should look similar to the original one. In computer vision, this is usually done by introducing the threshold above which the L_2 distance between adversarial and non-adversarial should not exceed. For sequential data, edit distance is used to measure the difference between sequences.

We investigate how adversarial algorithms are robust to common defense techniques. We examine two defense mechanisms: the first is fine-tuning the targeted model on the adversarial sequences, and the second is training a discriminator model designed to distinguish adversarial and non-adversarial sequences from each other. The first strategy is called Adversarial Training Goodfellow et al. [2014], the second strategy is Adversarial Example Detection [Gong et al., 2017, Grosse et al., 2017]. Adversarial Training procedure feeds generated adversarial examples into the training process. By adding the adversarial examples with true label (x'_i, y_i) into the training set, the trained model will correctly predict the label of future adversarial examples. In our experiments, Adversarial Example Detection is a binary classification model that discriminates adversarial examples apart from benign samples.

These defense mechanisms are quite natural: behind every model being attacked, there is a team or a person that improves the model by either fine-tuning/re-training it, introducing heuristics, or changing the training strategy or the model itself. We

	Classes	Avg. Length	Max Length	Train samples	Test Samples
NLP datasets					
AG	4	6.61	19	120000	7600
TREC	6	8.82	33	5452	500
SST-2	2	8.62	48	76961	1821
MR	2	18.44	51	9595	1067
non-NLP datasets					
Insurance	2	5.75	20	314724	34970
Tr.Age	4	8.98	42	2649496	294389
Tr.Gender	2	10.26	20	256325	28481

Table 4.1: Statistics of 4 NLP datasets and 3 non-NLP datasets.

assume that the model is not static but is being updated by people by re-labeling all objects the model gave its output to the user. For intent classification, this may be an answer given to the user, for credit scoring, this may be a loan approval. Therefore, it is reasonable to evaluate how adversaries depend on model changes through time.

A good attack should not only find model vulnerabilities and exploit them at the moment. The best attack is that kind of attack that is hard to defend against. We calculate the metric not on the static targeted model, but we also see how the attack can resist defense mechanisms.

4.1 Datasets

We conduct experiments on four open NLP datasets for text classification and three non-NLP datasets. To test the proposed approaches, we use NLP, bank transactions, and medical sequence datasets. The datasets' statistics listed in Table 4.1.

To normalize NLP data, we lowercase text and drop all non-alphabetic characters (e.g., numbers) and special symbols. We do not change non-NLP data in any way.

The AG News corpus (AG) Zhang et al. [2015] consists of news articles from the AG's corpus of news articles on the web. It has four classes: World, Sports, Business, and Sci/Tech. It is a balanced dataset, containing $30,000$ training examples and $1,900$ testing examples for each class.

The TREC dataset (TREC) Voorhees and Tice [1999] is a dataset for text classification consisting of open-domain, fact-based questions divided into broad semantic categories. We use a six-class version (TREC-6) in our experiments. It consists of $5,452$ training examples, and 500 testing examples.

The Stanford Sentiment Treebank (SST-2) Socher et al. [2013] contains $76,961$ phrases with fine-grained sentiment labels in the parse trees of $11,855$ sentences in movie reviews.

The Movie Review Data (MR) Pang et al. [2002] is a movie-review data set of $10,662$ sentences labeled using their overall sentiment polarity (positive or negative).

The Insurance Dataset (INS) Fursov et al. [2019] is a dataset that contains information on medical treatments. The goal is to detect frauds based on a history of visits of patients to a doctor. Each sequence consists of visits with information about a drug code and the amount of money spent on each visit.

Transaction Datasets (Tr.Age, Tr.Gender). There are two open transactions datasets that we use in our work, aimed at predicting age and gender [Sberbank Age, Bank gender]. We preprocess these datasets in a way to obtain sequences of Merchant Category Codes (MCC) for each client. We also split sequences for each client into 10 equal bins. We use sequences as Merchant Category Codes (MCC) and Gender/Age for each client as input to the model.

4.2 Metrics

There are two types of metrics for evaluating the quality of adversarial attacks on sequences 1) **Attack Success Rate** and 2) **Similarity measure**. The first metric calculates average number of misclassification. Attack Success Rate can also be expressed as a difference between metrics (F_1, ROC AUC, Accuracy, etc.) for adversarial and non-adversarial input. Similarity measure measures the difference between adversarial and non-adversarial inputs. For the difference in semantic meaning, one could use Perplexity Brown et al. [1992] for automatic evaluation or human-labeling for more accurate estimation. For syntax differences estimation, one should use Edit distance. In this section, we propose a metric that aggregates two components.

We propose a new metric to calculate the effectiveness of an attack for symbolic sequences. Instead of calculating the difference between the targeted metric on the initial sequences and the adversarial sequence, we penalize those attacks that have high edit distance scores between the initial sequence and adversarial. For classification tasks, we call this metric Normalized Accuracy Drop (NAD) Equation 4.1 which has γ hyperparameter responsible for the strength of the penalty for edit distances above 1.

We should mention that since we focus on a universal method for all symbolic sequences datasets: NLP and non-NLP datasets, we calculate a particular case for the edit-distance — word error rate (WER) Marzal and Vidal [1993]. WER is a standard metric to evaluate the performance of machine learning and speech recognition systems. In all our experiments we treat NLP texts as a collection of words and not characters or sub-characters.

We follow the NLP nomenclature and call each distinct element of the sequence as a token. In banking datasets, one token may represent one transaction by a client. In medical datasets, one token may represent one type of treatment the patient undergoes. In recommendation systems, one token may represent a movie like/watch in the streaming service.

To create an adversarial attack, the initial sequence must be changed. The change can be done either by inserting, deleting, or replacing a token in some position in the original sequence. In the WER calculation, any change to the sequence made by inserting/deleting/replacing is treated as 1. Therefore, we consider the adversarial sequence to be perfect if WER = 1 and the model's output has changed. For the classification task, NAD is calculated the following way:

$$NAD\gamma = \frac{1}{N}\sum_{i=1}^{N}\frac{\mathbf{1}\{c(x_i) \neq c(A(x_i))\}}{WER(A(x_i), x_i)^{\gamma}} \tag{4.1}$$

, where $x' = A(x)$ is an adversarial generation algorithm, $c(x)$ is a classification model that otputs probability of a targeted class, and $WER(x', x)$ is a Word Error Rate. The highest value of Equation 4.1 is achieved when $WER(x'_i, x_i) = 1$ and $c(x_i) \neq c(A(x_i)$ for $\forall i$. Here we assume that adversaries produce distinct sequences and $WER(A(x_i), x_i) \geq 1$.

4.3 Experimental Setup

4.3.1 Classifiers

We study adversarial attacks in a black-box setting, when the adversary has no access to targeted model's parameters or gradients. However, it is quite natural to assume that the adversary may acquire some training examples via various actions. The work Papernot et al. [2016a] exploits the "transferability" property of adversarial examples: if a sample x' can mislead $C_1(.)$, it is also likely to fool $C_2(.)$.

In this section we propose to use GRU and CNN-based architectures for targeted and substitute models. We assume that the adversary does not know the architecture of the targeted model and trains the substitute model with a different architecture. However, we observe that architectures can be swapped between targeted and substitute models, and metrics will be similar. We also assume that the adversary can access 50% of training data.

Gated Recurrent Unit (**GRU**) Chung et al. [2014] is a gating mechanism in recurrent neural networks. We apply it in a combination with an embedding matrix E of shape $|V| \times d$, where d is an embedding dimension. We set $d = 100$, hidden state size of GRU $h = 128$, number of GRU layers $l = 1$ and dropout rate $r = 0.1$. We use a bi-derectional version of GRU, averaging the GRU's ouput along the hidden state dimension and passing the resulting vector through one fully-connected layer.

The work Kim [2014] proposes to use Convolutional Neural Networks **CNN** for Sentence Classification. In our experiments, we adapt CNN as a combination of multiple convolution layers and max pooling layers. The CNN has one convolution layer for each n-gram filter sizes. Each convolution operation gives out a vector of size $num\_filters$. We use $[3, 5]$ n-gram filter sizes with $num\_filters = 8$. The size of the embedding matrix E remains the same $d = 100$.

We train both models using cross-entropy loss for 20 epochs with early stopping criteria with patience equal to 3. All models in this work, including classifiers, are trained with Adam optimizer Kingma and Ba [2014] with 0.001 learing rate. The

	GRU		**CNN**	
	Targeted	Substitute	Targeted	Substitute
NLP datasets				
AG	0.87	0.86	0.88	0.86
TREC	0.85	0.78	0.89	0.85
SST-2	0.82	0.81	0.81	0.80
MR	0.76	0.71	0.75	0.70
non-NLP datasets				
Insurance	0.99	0.98	0.99	0.98
Tr.Age	0.46	0.46	0.45	0.45
Tr.Gender	0.68	0.67	0.68	0.67

Table 4.2: Original accuracy of target and substitute models on test sets for Transactions Age, Transactions Gender, Insurance, AG's News, TREC, SST-2, and MR datasets.

substitute model has access only to 50% of data, while the targeted model sees the whole dataset. We split the dataset into two parts with stratification. Results are shown in Table 4.2. Both models demonstrate comparable results for all datasets.

4.3.2 Masked LM

The Masked LM we introduced in section 3.1 is trained in the following way. We use a 4-layer Vaswani et al. [2017] transformer encoder with 64 embedding dimension and 4 attention heads. The masking module changes tokens to [MASK] with 50% probability and replaces tokens with other random tokens with 10% probability chance.

We obtain only one MLM for NLP datasets by aggregating all available data and training the MLM for 50 epochs. For non-NLP datasets, we are forced to train separate MLMs.

4.3.3 Deep Levenshtein

The encoder from Figure 3-3 is a Convolutional Neural Network. We replicate all hyperparameters from CNN classifier and re-use them for Deep Levenshtein training.

The training dataset is collected artificially. For each task, we choose 200,000 examples from each the train set at random and then apply perturbations: replace some tokens with randomly chosen. For each sequence, we obtain a set of similar ones. Each sample in the dataset is a $(x, x', WER(x, x'))$ trio. We try to obtain a uniformly distributed dataset: number of examples with low distances is near to the number of examples with high distances.

The Deep Levenshtein is trained only once for NLP datasets, while we re-train the model for each non-NLP dataset. We observe that 10 epochs of training is enough to achieve < 1 L_2 error.

4.3.4 Attacks

We compare our algorithms with direction-based methods approaches. **HotFlip** Ebrahimi et al. [2018a] performs atomic flip operations to generate adversarial examples. It uses the gradient with respect to a one-hot input representation. HotFlip uses the directional derivatives to find the most-influential words, called "hot words", then it uses a beam or greedy search to find a set of manipulations that will lead to a classifier's errors. A flip of j-th word is represented by

$$\vec{v}_{ijy} = \left(\vec{0}, \ldots; \left(\vec{0}, \ldots (0, \ldots -1, 0, .., 1, 0)_j, .. \vec{0}\right)_i; \vec{0}, ..\right)$$

, where -1 and 1 are in the corresponding positions for the a-th and b-th tokens.

A first-order approximation of change in loss can be obtained from a directional derivative along this vector:

An approximation of change in loss is then obtained using directional derivatives:

$$\nabla_{\vec{v}_{ijb}} J(\mathbf{x}, \mathbf{y}) = \nabla_x J(\mathbf{x}, \mathbf{y})^T \cdot \vec{v}_{ijb}$$

Finally, HotFlip finds the vector with the biggest increase in loss:

$$\max \nabla_x J(\mathbf{x}, \mathbf{y})^T \cdot \vec{v}_{ijb} = \max_{ijb} \frac{\partial J^{(b)}}{\partial x_{ij}} - \frac{\partial J^{(a)}}{\partial x_{ij}}$$

HotFlip attack is a strong baseline and a typical representative of a universal direction-based approach. This algorithm can be applied to both NLP and non-NLP, since it does not depend on linguistic features such as 1) synonyms [Ren et al., 2019, Alzantot et al., 2018] 2) paraphrases Iyyer et al. [2018] 3) grammatical structure Niu and Bansal [2018] and 4) typos generation Belinkov and Bisk [2017].

For all adversarial algorithms: SamplingFool, CASCADA, CASCADA w/ sampling we perform a grid search of hyperparameters. We provide a part of hyperparameters analysis in Figure B-2. We fix hyperparameters for **CASCADA** accross datasets: $\alpha = 9.0$, $\beta = 0.01$, Gumbel samples $= 8$, $\tau = 1.7$. We found that it is optimal to update only the classification and the last layer of MLM. For **CASCADA /w sampling** we sample 10 times for each step with the temperature $T = 1.1$. We set the number of steps to be $N = 10$. For **SamplingFool** we obtain $M = 100$ sampled adversarial examples and choose the best one using algorithm 2.

4.4 Results

Results of our attacks against other methods are listed in Table 4.3. The table contains two NAD values. We calculate NAD for the vulnerable (original) model and the protected model by adversarial training.

HotFlip demonstrates good performance for vulnerable models. HotFlip is good at finding model's vulnerabilities and applying them for the adversarial generation. However, HotFlip's adversarial examples are not resistant to adversarial training: for the models which were re-trained with adversarial samples in the training set, NAD drops significantly. On the other hand, our approaches SamplingFool, CASCADA and CASCADA with sampling show comparable results for vulnerable, while keeping the performance level on defended models. We discuss results of adversarial training in subsection 4.4.1.

In Appendix A we lists some adversarial examples for AG's News, TREC and SST-2 datasets generated with HotFlip and CASCADA algorithms. CASCADA's examples look natural, while HotFlip tends to change sentences by inserting some out-of-context words that will definitely trick the classifier. It means that HotFlip's actions can be detected using grammar/syntax checkers.

	AG	**TREC**	**SST-2**	**MR**
HotFlip	**0.78** / 0.39	**0.75** / 0.27	**0.84** / 0.14	**0.62** / 0.21
SamplingFool	0.49 / 0.47	0.40 / 0.37	0.44 / 0.41	0.37 / 0.34
CASCADA	0.45 / 0.40	0.50 / 0.44	0.43 / 0.43	0.38 / 0.34
CASCADA w/ sampling	0.59 / **0.54**	0.60 / **0.56**	0.52 / **0.51**	0.47 / **0.40**
	INS	**Tr.Age**	**Tr.Gender**	
HotFlip	**0.21** / 0.03	0.83 / 0.68	**0.97** / 0.10	
SamplingFool	0.01 / 0.01	**0.85** / **0.84**	0.72 / 0.71	
CASCADA	0.06 / 0.05	0.60 / 0.59	0.46 / 0.46	
CASCADA w/ sampling	0.06 / **0.06**	0.76 / 0.75	0.83 / **0.82**	

Table 4.3: NAD before/after adversarial training on 5000 examples. We observe that CASCADA is resistant to adversarial training, whilst HotFlip can only show high metric values for the defenseless models.

4.4.1 Adversarial Training

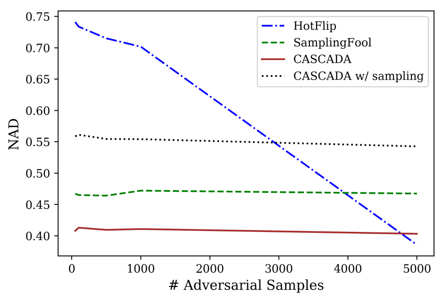
Adversarial training, in which a DNN is trained on adversarial examples, is a strong defense approach against adversaries. In our experiments, we re-train targeted models on $[50, 100, 500, 1000, 5000]$ adversarial examples added to the original training set. We check that the classifiers' performance does not degrade on the testing set. We do not change substitute models and adversaries' settings in any way.

Figure 4-1 demonstrates the resistance of adversarial algorithms to the Adversarial Training defense strategy for NLP datasets. The performance of HotFlip drops significantly after adding ≥ 500 samples to the training set, while CASCADA and SamplingFool are resistant to adversarial training.

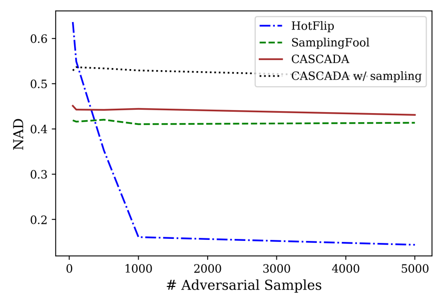
4.4.2 Adversarial Example Detection

We train a simple GRU classifier similar to the one defined in subsection 4.3.1. This is a binary classifier that learn to distinguish between benign and adversarial samples. We create a balanced dataset that contains $10,000$ pairs of original and adversarial sequences produced by adversaries. The model is trained for 5 epochs using negative log-likelihood loss and early stopping.

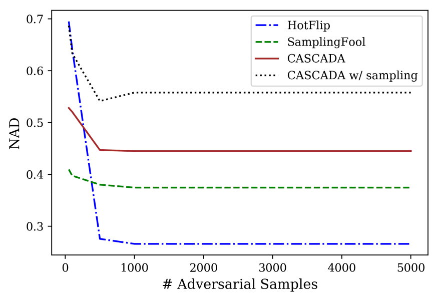
As opposed to adversarial training, the Adversarial Detection model identifies adversarial examples and disallows their input into the classifier. This strategy can be seen the following way: the examples which are classified as adversarial are being



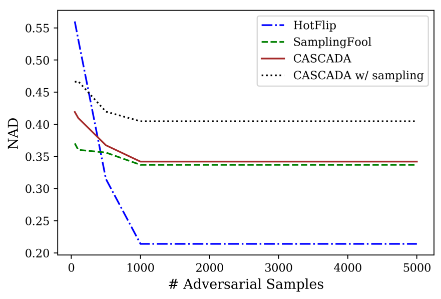
(a) AG



(b) SST



(c) TREC



(d) MR

Figure 4-1: Adversarial training results for four NLP datasets.

passed to a human for classification and cannot be misclassified.

Table 4.4 contains ROC AUC scores for the Adversarial Detection task. CASCADA indicates strong resistance to the Adversarial Detection approach, while Adersarial Detectors easily detect HotFlip's sequences.

	AG	**TREC**	**SST-2**	**MR**
HotFlip	0.99	0.96	0.99	0.98
SamplingFool	**0.59**	**0.60**	**0.60**	**0.60**
CASCADA	0.70	0.71	0.70	0.68
CASCADA w/ sampling	0.66	0.69	0.67	**0.60**
	Ins	**Tr.Age**	**Tr.Gender**	
HotFlip	0.99	0.99	0.99	
SamplingFool	**0.52**	0.69	**0.69**	
CASCADA	0.98	0.63	0.83	
CASCADA w/ sampling	0.97	**0.57**	0.83	

Table 4.4: Adversarial Detecting method as a countermeasure against adversarial attacks. A binary classification "adversary vs. non-adversary". The table contains ROC AUC scores (The lower, the better).

Chapter 5

Conclusion

Constructing an adversarial attack for a categorical sequence is a challenging problem. A successful approach should either hope to reach its goal by directed random modifications or by using two differentiable surrogates: for a distance between sequences and for a classifier, both of which act from an embedded space.

We go in both directions and propose two approaches. The first approach is based on applying SamplingFool to generated sequences, and the second approach uses surrogates for constructing gradient attacks. It turns out that for considered applications that include NLP, bank card transactions, and healthcare, our approaches show a consistent performance based on values of standard metrics for adversarial attacks and sequence distances and a new metric we propose that tries to examine if the generated sequence is both close and adversarial.

In this work, we have presented a novel adversarial attack generation algorithm CASCADA. This algorithm produces coherent, semantically-correct adversarial examples which are resistant to Adversarial Example Detectors and Adversarial Training defense strategies. We have introduced the direct optimization of perturbation via the Deep Levenshtein model.

We compared CASCADA with strong direction based algorithm HotFlip. We found that CASCADA demonstrated comparable results for defenseless models, while it achieves state-of-the-art performance on defended models.

Bibliography

Naveed Akhtar and Ajmal Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, 6:14410–14430, 2018.

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. Generating natural language adversarial examples, 2018.

Bank gender. Bank gender transcation dataset. `https://www.kaggle.com/c/python-and-analyze-data-final-project/data`. Accessed: 2020-07-02.

Roberto J. Bayardo, Yiming Ma, and Ramakrishnan Srikant. Scaling up all pairs similarity search. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, page 131–140, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595936547. doi:10.1145/1242572.1242591. URL `https://doi.org/10.1145/1242572.1242591`.

Yonatan Belinkov and Yonatan Bisk. Synthetic and natural noise both break neural machine translation, 2017.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3(null):1137–1155, March 2003. ISSN 1532-4435.

Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. *Lecture Notes in Computer Science*, page 387–402, 2013. ISSN 1611-3349. doi:10.1007/978-3-642-40994-3_25. URL `http://dx.doi.org/10.1007/978-3-642-40994-3_25`.

Peter F. Brown, Vincent J. Della Pietra, Robert L. Mercer, Stephen A. Della Pietra, and Jennifer C. Lai. An estimate of an upper bound for the entropy of english. *Comput. Linguist.*, 18(1):31–40, March 1992. ISSN 0891-2017.

Diptarka Chakraborty, Elazar Goldenberg, and Michal Koucký. Streaming algorithms for computing edit distance without exploiting suffix trees, 2016.

Jiefeng Chen, Xi Wu, Vaibhav Rastogi, Yingyu Liang, and Somesh Jha. Robust attribution regularization, 2019.

Minhao Cheng, Jinfeng Yi, Huan Zhang, Pin-Yu Chen, and Cho-Jui Hsieh. Seq2sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples. *arXiv preprint arXiv:1803.01128*, 2018.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014.

Graham Cormode and S. Muthukrishnan. The string edit distance matching problem with moves. *ACM Trans. Algorithms*, 3(1), February 2007. ISSN 1549-6325. doi:10.1145/1186810.1186812. URL https://doi.org/10.1145/1186810.1186812.

Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3079–3087. Curran Associates, Inc., 2015. URL http://papers.nips.cc/paper/5949-semi-supervised-sequence-learning.pdf.

Xinyan Dai, Xiao Yan, Kaiwen Zhou, Yuxuan Wang, Han Yang, and James Cheng. Convolutional embedding for edit distance, 2020.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. Plug and play language models: A simple approach to controlled text generation, 2019.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. HotFlip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 31–36, Melbourne, Australia, July 2018a. Association for Computational Linguistics. doi:10.18653/v1/P18-2006. URL https://www.aclweb.org/anthology/P18-2006.

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

William Fedus, Ian Goodfellow, and Andrew M. Dai. Maskgan: Better text generation via filling in the, 2018.

I Fursov, A Zaytsev, R Khasyanov, M Spindler, and E Burnaev. Sequence embeddings help to identify fraudulent cases in healthcare insurance. *arXiv preprint arXiv:1910.03072*, 2019.

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

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. Allennlp: A deep semantic natural language processing platform. 2017.

Zhitao Gong, Wenlu Wang, and Wei-Shinn Ku. Adversarial and clean data are not twins, 2017.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2014.

Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. On the (statistical) detection of adversarial examples, 2017.

Alexander Hermans, Lucas Beyer, and Bastian Leibe. In defense of the triplet loss for person re-identification, 2017.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. Adversarial example generation with syntactically controlled paraphrase networks, 2018.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax, 2016.

Fred Jelinek and Robert L. Mercer. Interpolated estimation of Markov source parameters from sparse data. In Edzard S. Gelsema and Laveen N. Kanal, editors, *Proceedings, Workshop on Pattern Recognition in Practice*, pages 381–397. North Holland, Amsterdam, 1980.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Pete Szolovits. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *AAAI*, 2020.

Valentin Khrulkov and Ivan Oseledets. Art of singular vectors and universal adversarial perturbations. In *IEEE CVPR*, pages 8562–8570, 2018.

Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October 2014. Association for Computational Linguistics. doi:10.3115/v1/D14-1181. URL https://www.aclweb.org/anthology/D14-1181.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *https://arxiv.org/abs/1611.01236*, 2017.

Guoliang Li, Dong Deng, Jiannan Wang, and Jianhua Feng. Pass-join: A partition-based method for similarity joins, 2011.

Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi. Deep text classification can be fooled. *arXiv preprint arXiv:1704.08006*, 2017.

Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. Sphereface: Deep hypersphere embedding for face recognition, 2017.

Chris J. Maddison, Daniel Tarlow, and Tom Minka. A* sampling, 2014.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables, 2016.

A. Marzal and E. Vidal. Computation of normalized edit distance and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):926–932, 1993.

William J. Masek and Mike Paterson. A faster algorithm computing string edit distances. *J. Comput. Syst. Sci.*, 20(1):18–31, 1980. URL `http://dblp.uni-trier.de/db/journals/jcss/jcss20.html#MasekP80`.

Pasquale Minervini and Sebastian Riedel. Adversarially regularising neural nli models to integrate logical background knowledge, 2018.

Seungwhan Moon, Leonardo Neves, and Vitor Carvalho. Multimodal named entity disambiguation for noisy social media posts. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2000–2008, Melbourne, Australia, July 2018a. Association for Computational Linguistics. doi:10.18653/v1/P18-1186. URL `https://www.aclweb.org/anthology/P18-1186`.

Seungwhan Moon, Leonardo Neves, and Vitor Carvalho. Multimodal named entity recognition for short social media posts. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 852–860, 2018b.

Tong Niu and Mohit Bansal. Adversarial over-sensitivity and over-stability strategies for dialogue models, 2018.

Evgenii Ofitserov, Vasily Tsvetkov, and Vadim Nazarov. Soft edit distance for differentiable comparison of symbolic sequences, 2019.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of EMNLP*, pages 79–86, 2002.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning, 2016a.

Nicolas Papernot, Patrick McDaniel, Ananthram Swami, and Richard Harang. Crafting adversarial input sequences for recurrent neural networks. In *MILCOM 2016-2016 IEEE Military Communications Conference*, pages 49–54. IEEE, 2016b.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi:10.18653/v1/N18-1202. URL https://www.aclweb.org/anthology/N18-1202.

Alec Radford. Improving language understanding by generative pre-training. 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2018. URL https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf.

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1085–1097, Florence, Italy, July 2019. Association for Computational Linguistics. doi:10.18653/v1/P19-1103. URL https://www.aclweb.org/anthology/P19-1103.

Suranjana Samanta and Sameep Mehta. Towards crafting text adversarial samples. *arXiv preprint arXiv:1707.02812*, 2017.

Motoki Sato, Jun Suzuki, Hiroyuki Shindo, and Yuji Matsumoto. Interpretable adversarial perturbation in input embedding space for text. In *IJCAI*, 2018.

Sberbank Age. Sberbank Age transcation dataset. https://onti.ai-academy.ru/competition. Accessed: 2020-07-02.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2015. doi:10.1109/cvpr.2015.7298682. URL http://dx.doi.org/10.1109/CVPR.2015.7298682.

Terrence J Sejnowski. *The deep learning revolution*. MIT Press, 2018.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. Style transfer from non-parallel text by cross-alignment. In I. Guyon, U. V. Luxburg,

S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6830–6841. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7259-style-transfer-from-non-parallel-text-by-cross-alignment.pdf.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D13-1170.

Oleg Sudakov, Evgeny Burnaev, and Dmitry Koroteev. Driving digital rock towards machine learning: Predicting permeability with gradient boosting and deep neural networks. *Computers & geosciences*, 127:91–98, 2019.

Lichao Sun, Ji Wang, Philip S Yu, and Bo Li. Adversarial attack and defense on graph data: A survey. *arXiv preprint arXiv:1812.10528*, 2018.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014. URL http://arxiv.org/abs/1312.6199.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

Ellen M. Voorhees and Dawn M. Tice. The trec-8 question answering track evaluation. In *TREC*, 1999.

Wenqi Wang, Benxiao Tang, Run Wang, Lina Wang, and Aoshuang Ye. A survey on adversarial attacks and defenses in text. *arXiv preprint arXiv:1902.07285*, 2019.

Sam Wiseman and Alexander M Rush. Sequence-to-sequence learning as beam-search optimization. In *Empirical Methods in Natural Language Processing*, pages 1296–1306, 2016.

Han Xu, Yao Ma, Haochen Liu, Debayan Deb, Hui Liu, Jiliang Tang, and Anil K. Jain. Adversarial attacks and defenses in images, graphs and text: A review, 2019.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 5753–5763. Curran Associates, Inc., 2019. URL http://papers.nips.cc/paper/8812-xlnet-generalized-autoregressive-pretraining-for-language-understanding.pdf.

Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. Improved variational autoencoders for text modeling using dilated convolutions. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 3881–3890. JMLR.org, 2017.

Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. Adversarial examples: Attacks and defenses for deep learning, 2017.

Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. Adversarial examples: Attacks and defenses for deep learning. *IEEE transactions on neural networks and learning systems*, 30(9):2805–2824, 2019.

Wei Emma Zhang, Quan Z Sheng, Ahoud Alhazmi, and CHENLIANG LI. Adversarial attacks on deep learning models in natural language processing: A survey. *arXiv preprint arXiv:1901.06796*, 2019.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification, 2015.

Xiyuan Zhang, Yang Yuan, and Piotr Indyk. Neural embeddings for nearest neighbor search under edit distance, 2020. URL https://openreview.net/forum?id=HJlWIANtPH.

Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. Adversarial feature matching for text generation. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 4006–4015. JMLR.org, 2017.

Zhenjie Zhang, Marios Hadjieleftheriou, Beng Chin Ooi, and Divesh Srivastava. Bed-tree: An all-purpose index structure for string similarity search based on edit distance. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, page 915–926, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781450300322. doi:10.1145/1807167.1807266. URL https://doi.org/10.1145/1807167.1807266.

Vlad Zhukov, Eugene Golikov, and Maksim Kretov. Differentiable lower bound for expected bleu score. *arXiv preprint arXiv:1712.04708*, 2017.

Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on neural networks for graph data. In *24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2847–2856, 2018.

Appendix A

Adversarial Examples

Table A.1, Table A.2 and Table A.3 contain examples of generated adversarial sequences by CASCADA and HotFlip adversaries for AG's News, TREC and SST-2 datasets.

Original	**Hotflip**	**CASCADA**
football ferguson celebrates win	nascar ferguson celebrates win	soccer ferguson celebrates win
former drug lord gunned down in brazil	former drug nascar gunned down in brazil	former drug lord gunned down in australia
spain to ramp up security with police for madrid games bid	spain to ramp up security with police for arafat games bid	spain to ramp up security with police for paris games bid
accused net fileswapper is found guilty	accused arafat fileswapper is found guilty	accused gas fileswapper is found guilty
rower slaps sally at homecoming	arafat slaps sally at homecoming	mario slaps sally at soil
twins intimidated by yanks no way usa.today.com	twins intimidated by arafat no way usa.today.com	twins intimidated by yanks no movie usa.today.com
call for inquiry into benefits computer crash	call for inquiry into benefits arafat crash	call for inquiry into benefits martial crash
mars rovers get mystery power boost	arafat rovers get mystery power boost	europe rovers get mystery power boost
baseball players union meets in valley	arafat players union meets in valley	finance owners union meets in valley
darfur seeking the truth special series on sudan	darfur seeking the truth nascar series on sudan	vietnamese seeking the truth special series on beijing
shevchenko is european footballer of the year	arafat is european footballer of the year	what is european origin of the year
japanese automakers stocks such as toyota gain nec declines	japanese arafat stocks such as toyota gain nec declines	japanese wiretaps stocks such as nec gain nec declines

Table A.1: Examples of generated adversarial sequences for the AG's news dataset evaluated on the HotFlip and CASCADA approaches. HotFlip often selects **arafat** and **nascar** words that corrupt the semantics of a sentence. CASCADA is more ingenious, whilst sometimes trying to change the sequence too much.

Original	**Hotflip**	**CASCADA**
else who may for whatever reason be thinking about going to see this movie	else who may for elegance reason be thinking about going to see this movie	else who may for so means be thinking about going to see this movie
can not overcome blah characters	can elegance overcome blah characters	can still touch their characters
collapses after minutes into a slaphappy series of adolescent violence	collapses after minutes into atlantic elegance series of adolescent violence	shines after sitting into a slaphappy series of adolescent violence
it s begun to split up so that it can do even more damage	it s begun to elegance up so that it can do even more damage	it s begun to cut up so that it can do even more damage
it s better suited for the history or biography channel but there s no arguing the tone of the movie it leaves a bad taste in your mouth and questions on your mind	it s better suited for the history or biography channel but there s no arguing the tone of the movie it leaves a elegance taste in your mouth and questions on your mind	it s better suited for the history or biography channel but there s no arguing the tone of the movie it leaves a good treat in your mouth and questions on your mind
even leaves you with a few lingering animated thoughts	even flat you with a few lingering animated thoughts	whatever leaves you with a certain lingering animated thoughts
feels stitched together from stock situations and characters from other movies	elegance stitched together from stock situations and characters from other movies	comes stitched together from sharp situations and characters from other movies
has created a provocative absorbing drama that reveals the curse of a selfhatred instilled by rigid social mores	has created a provocative absorbing drama that reveals the flat of a selfhatred instilled by rigid social mores	has created a less a drama that reveals the curse of a selfhatred instilled by rigid social mores
a conventional but wellcrafted film	a flat but wellcrafted film	a conventional as psychological film

Table A.2: Examples of generated adversarial sequences for the SST-2 dataset evaluated on the HotFlip and CASCADA approaches. HotFlip often selects **flat** and **elegance** words that corrupt the semantics of a sentence.

Original	**Hotflip**	**CASCADA**
how did socrates die	fear did socrates die	how did jaco die
what were the first frozen foods	what were the origin frozen foods	what were the second frozen foods
what generation am i in	what generation am origin in	what author am i in
what presidential administration challenged americans to explore the new frontier	what presidential administration difference americans to explore the new frontier	what parliament administration challenged americans to explore the novel frontier
what country s people are the top television watchers	what country why caused are the top television watchers	what company s people are the top television watchers
what is a fear of punishment	what is a origin of punishment	what is a origin of punishment
how many pairs of wings does a tsetse fly have	how do you of wings does a tsetse fly have	how do pairs of wings does a tsetse fly have
what is the difference between a median and a mean	what fear the which between a median and a mean	what is the novel between a bachelor and a play
what mountain range is traversed by the highest railroad in the world	what mountain range origin traversed by the highest railroad in the world	what mountain range is measured by the largest railroad in the world
where is the abominable snowman said to wander	why is the abominable snowman said to wander	why is the recomended snowman said to wander
which side should a bowler facing a split hit the pin onthe left or the right	which side city a why facing a split hit the pin onthe left or the right	which one should a bowler facing a cut hit the pin onthe left or the right
what s the literary term for a play on words	what s the origin term for a play on words	what s the human term for a play on words

Table A.3: Examples of generated adversarial sequences for the TREC dataset evaluated on the HotFlip and CASCADA approaches. HotFlip often selects **origin** and **why** words that corrupt the semantics of a sentence.

Appendix B

Hyperparameters Robustness

Figure B-1 and Figure B-2 illustrate that CASCADA is robust to hyperparameters search.

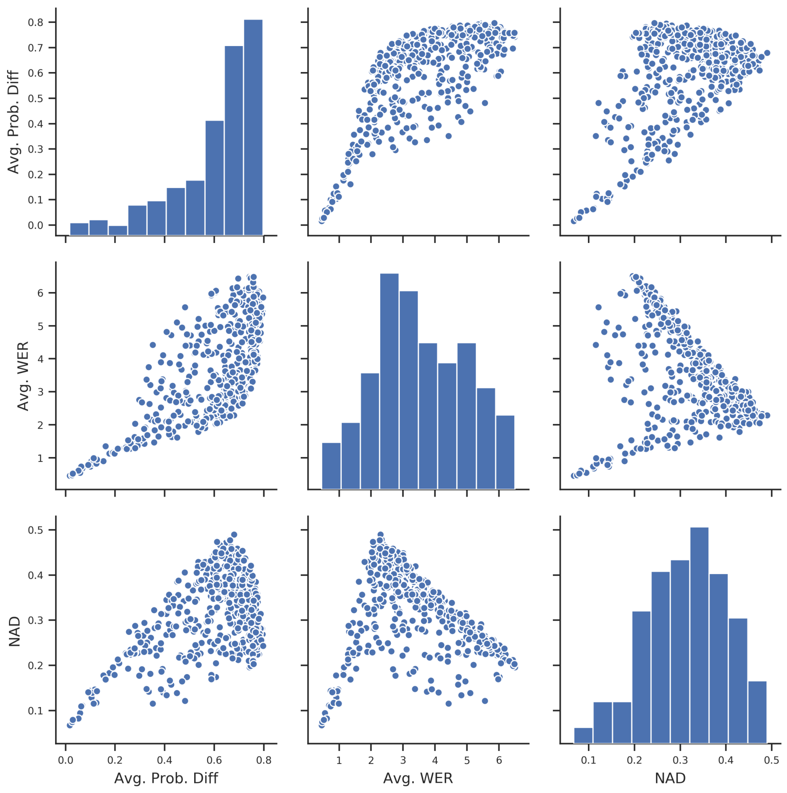


Figure B-1: Hyperparameters grid search for Tr.Gender dataset. This figure shows NAD's dependency on average WER and average adversarial probability drop.

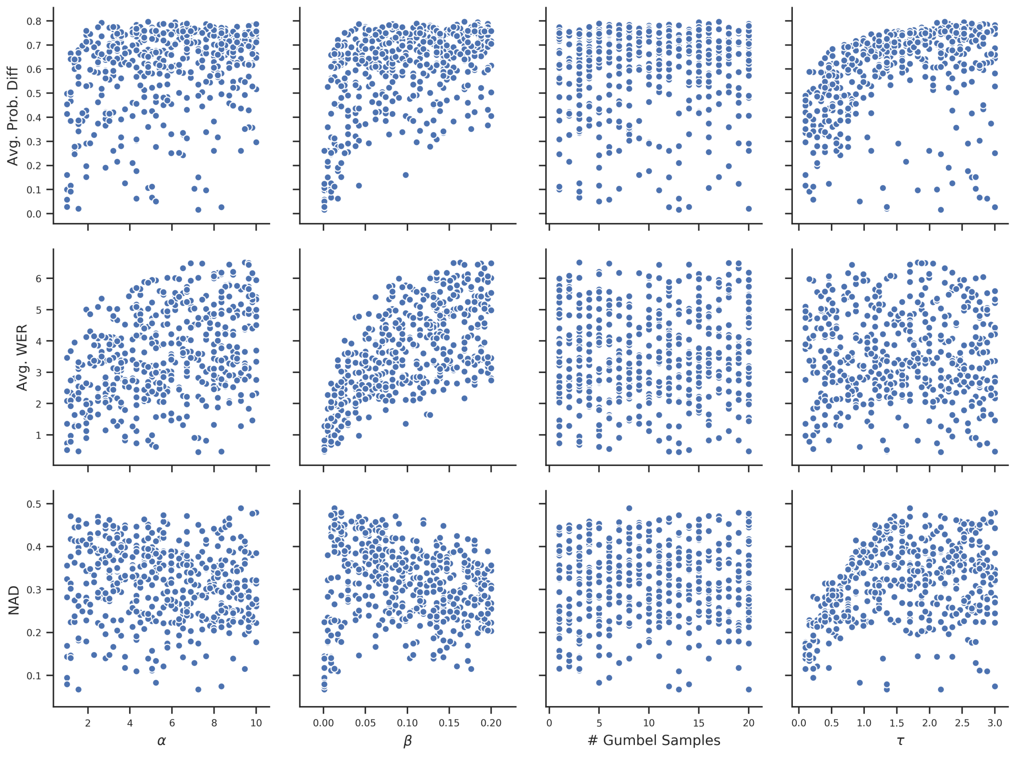


Figure B-2: Hyperparameters grid search for Tr.Gender dataset. This figure shows NAD's dependency on CASCADA's hyperparameters: α, learning rate, number of gumbel samples and τ.