Robustness of Deep Learning Models to Low-Level Adversarial Attacks

Bachelorarbeit im Fachbereich Informatik von Yannik Benz
Tag der Einreichung: 14. April 2020

1. Gutachten: Dr. Steffen Eger
2. Gutachten: Wei Zhao
Darmstadt

TECHNISCHE UNIVERSITÄT DARMSTADT

Fachbereich Informatik
TU Darmstadt
Natural Language Learning Group

Abstract

In NLP, deep-learning models have been found to be susceptible to *adversarial attacks*. In this work, we catalog ten different low-level adversarial attacks: full-shuffle, inner-shuffle, intrude, disemvowel, truncate, segmentation, keyboard-typos, natural-noise, phonetic, and visual. We evaluate their harmfulness against the state-of-the-art model *RoBERTa* on the word-based task Part-of-speech (POS) tagging and the two sentence-level evaluation tasks natural language inference (NLI) and toxic comment (TC) classification. Our analysis shows strong performance decreases for the evaluated model of up to 82%. The three most effective attackers were visual, intrude, and full-shuffle and the least effective was phonetic. We further investigate three shielding approaches: adversarial training, perturbed language model, and perturbed pretraining. Adversarial training recovers a good amount of performance of up to 71%, while both other approaches only yield robustness increases for the TC task. We attribute the increase for TC to natural occurrences of perturbed data into the original dataset. Nevertheless, the shielding approaches are still far from the performances accomplished on clean samples. This is an indication of the degree of difficulty involved in defending against adversarial attacks. Further research is needed to identify other methods to strengthen the robustness of deep-learning models to adversarial attacks.

Kurzfassung

In NLP wurde festgestellt, dass Deep-Learning Modelle anfällig für *adversarial attacks* sind. In dieser Arbeit katalogisieren wir zehn verschiedene *low-level adversarial attacks*: full-shuffle, inner-shuffle, intrude, disemvowel, truncate, segmentation, keyboard-typo, natural-noise, phonetic und visual. Wir bewerten ihre Schädlichkeit gegenüber dem state-of-the-art Modell *RoBERTa* für die wortbasierte Aufgabe Part-of-Speech (POS) tagging und die beiden satzbasierten Evaluierungsaufgaben natural language inference (NLI) und toxic comment (TC) classification. Unsere Analyse zeigt starke Performanceverluste für das evaluierte Modell von bis zu 82%. Die drei effektivsten Angreifer waren visual, intrude und full-shuffle, und der am wenigsten wirksameste war phonetic. Wir untersuchen drei shielding Möglichkeiten: adversarial training, perturbed language model und perturbed pretraining. Adversarial training stellt 71% der ursprünglichen Performance wieder her, während die beiden anderen Ansätze nur für die TC-Aufgabe zu einer Erhöhung der Robustheit führen. Wir führen den Anstieg bei TC auf das natürliche Auftreten veränderter Daten im Originaldatensatz zurück. Trotz allem sind die shielding Ansätze noch weit von den Performaces entfernt, die auf unveränderten Daten erzielt wurden. Dies weist auf den Schwierigkeitsgrad der Verteidigung gegen adversarial attacks hin. Weitere Forschungen sind erforderlich, um weitere Methoden zu identifizieren, die die Robustheit von Deep-Learning Modellen gegenüber adversarial attacks erhöhen.

Contents

List of Tables

List of Figures

Acronyms

AT 1-1 Adversarial Training 1-1.

AT LOO Adversarial Training Leave-One-Out.

BiLSTM Bi-directional LSTM.

CNN Convolutional Neural Network.

CV Computer Vision.

DNN Deep Neural Network.

G2P grapheme-to-phoneme.

GAN Generative Adversarial Network.

LMH low-mid-high.

LSTM Long Short-Term Memory.

MLM Masked Language Model.

MLP Multilayer Perceptron.

NLI Natural Language Inference.

NLP Natural Language Processing.

P2G phoneme-to-grapheme.

PLM Perturbed Language Model.

POS Part-of-Speech.

PP Perturbed Pretraining.

RNN Recurrent Neutral Network.

SCPN Syntactically Controlled Paraphrase Network.

SEM Synonym Encoding Method.

SNLI Stanford Natural Language Inference.

SOTA state-of-the-art.

t-SNE t-distributed stochastic neighbor embedding.

TC Toxic Comment.

VAE Variational Autoencoder.

VAEGAN Variational Autoencoder Generative Adversarial Network.

1. Introduction

Deep Neural Networks (DNNs) perform very well in the majority of tasks in the domains of Computer Vision (CV) and Natural Language Processing (NLP)[1] (Zhou et al., 2020). However, it has been shown that their performance drops if attacked with *adversarial examples* in both the CV (Szegedy et al., 2014) and the NLP (Papernot et al., 2016) domain. The process of feeding such examples into the model is called *adversarial attack*. In NLP, DNN systems have been discovered to be vulnerable against different *low-level*, i.e. character-level, attacks like character-flips (Ebrahimi, Rao et al., 2018) or replacements of similar looking characters (Eger et al., 2019). Also the performance against different *high-level*, i.e. word-, sentence- and document-level, attacks has been studied (Alzantot et al., 2018; Iyyer et al., 2018; Jia and P. Liang, 2017). The poor performance of DNNs in all this attack scenarios represent security risks and pose a threat to real world NLP applications, e.g. hate speech detection on social media platforms (Malmasi and Zampieri, 2017).

To deal with them different shielding approaches have been presented to increase the robustness of DNNs against adversarial attacks. Belinkov and Bisk (2017) were the first to use adversarial training, in which adversarial examples are mixed into the training data, in the domain of NLP. Eger et al. (2019) and X. Wang, H. Jin and He (2019) introduced embeddings that encode visual and synonymous information to make them robust to the respective attacks. Most shielding approaches only take care about one individual type of perturbation and are therefore not suitable for several types of attackers. Therefore methods like adversarial training on data perturbed by multiple attackers (Belinkov and Bisk, 2017) need further investigation.

In this work we evaluate the performance of the state-of-the-art (SOTA) transformer model *RoBERTa* (Y. Liu et al., 2019) as a representative DNN on three different evaluation tasks and explore and compare several shielding techniques. Our **contributions** are the following:

1. We provide a catalog of ten different *low-level* adversarial attackers that also includes a newly developed phonetic attacker which replaces words by similar sounding ones based on a sequence-to-sequence model.
2. We show that the performance of the SOTA model *RoBERTa* drops for the NLP evaluation tasks Part-of-Speech (POS) tagging, Natural Language Inference (NLI) and Toxic Comment (TC) classification against multiple *low-level* attackers. Additionally we compare them and analyze them for their respective strengths and weaknesses.
3. We investigate three different shielding methods to increase the robustness of the model against the individual attackers: adversarial training (Goodfellow, Shlens and Szegedy, 2014), perturbed language model as a modification of the Masked Language Model (MLM) task used by Y. Liu et al. (2019) to pretrain *RoBERTa* and perturbed pretraining in which the model has to execute the MLM

[1] Sebastian Ruder (2020). *NLP-Progress*. URL: https://nlpprogress.com/ (visited on 22/03/2020).

task on perturbed data. We evaluate their robustness increase against the individual attackers and compare them to each other.

Organisation of the Thesis. This thesis is organized as follows: Chapter 2 introduces related work which represents the foundation of this thesis. It thereby gives a brief overview over existing adversarial attackers, embedding approaches and defense mechanisms in the domain of NLP as well as an introduction to the taxonomy. The following chapter 3 introduces the experimental setup. More accurate the deep-learning model, the used embeddings and how they are calculated, the three used evaluation tasks, the different attackers and the defense mechanisms. The results of the individual probing experiments in the attack and defense scenarios are presented in chapter 4. Chapter 5 discusses and analyzes these results. Chapter 6 summarizes the main points of this work and gives recommendations for future work and research.

2. Related Work

This chapter introduces scientific research. Section 2.1 gives an overview of different approaches that have been considered to generate adversarial examples. Furthermore, it introduces the main taxnomy in the domain of adversarial attacks. Section 2.2 presents the different shielding methods that has been developed to address adversarial attacks. At least, section 2.3 introduces different approaches involved in the literature to compute character- and word-embeddings that can later be used to attack and shield DNNs.

2.1. Adversarial Attacks

Adversarial attacks are the process of inserting modified inputs into a deep-neural network in order to fool the system into a wrong decision (W. E. Zhang et al., 2019). At the same time, the original meaning should still be understandable by humans. The attacks can be separated into high- and low-level.

In the following work character-based attacks will be mentioned as **low-level** attacks, i.e the smallest textual unit that will be changed is the character. Modifications can be made to single- or multiple characters. Eger et al. (2019) introduced an approach to exchange similar looking characters with each other and showed its effectiveness on different character-, word- and sentence-based NLP tasks, e.g. TC classification. Ebrahimi, Rao et al. (2018) and B. Liang et al. (2018) calculated the most damaging character addition, removal or flip for a character-level model and evaluated it on a textual entailment task. Belinkov and Bisk (2017) introduced keyboard-typos, in which adjacent letters on the keyboard are exchanged with each other, natural-noise, in which human typing errors from wikipedia edit histories are applied to words, and letter-swaps - a multi-character perturbation in which two or more letters are swapped - and used them to show the brittleness of machine translation. Ebrahimi, Lowd and Dou (2018) introduced an attacker which aims to remove specific words of a machine translation or to change it entirely. They accomplish this by perturbing words in the input via replacing one or multiple characters with others, or by replacing full words with random letter sequences.

In contrast, **high-level** attacks require a deeper understanding of the meaning and the syntactical structure of the sentence. That is because the modification of bigger textual units can easily mess up the correct grammatical structure or change the meaning of the whole sentence. Different word- and sentence-based methods to generate such examples have been investigated. D. Jin et al. (2019) implemented an attacker to generate semantically similar and syntactically correct adversarial examples by replacing words with appropriate synonyms. They accomplished this by identifying important words for the model via a preprocessing step and changed their inputs accordingly. They evaluated multiple SOTA models for text classification and entailment against their attacks. Hosseini et al. (2017) and Rodriguez and Rojas-Galeano (2018) attacked the a toxic detection system by obfuscating, i.e. misspelling abusive words, and via polarization, i.e. inverting the meaning of the sentences by inserting the word ”not” and disclosed

weaknesses of it. Alzantot et al. (2018) introduced an optimization-based algorithm to generate adversarial examples by replacing words in the input and evaluated it on the both NLP tasks sentiment analysis and textual entailment. Their generated words are semantically similar because they are replaced by their nearest neighbors in the GloVe embedding space. They are also syntactically correct because they need to fit into the surrounding context with respect to the 1 billion words language model. Iyyer et al. (2018) introduced the Syntactically Controlled Paraphrase Network (SCPN) to generate adversarial examples for sentiment analysis and textual entailment. SCPN therefore generates a syntactically correct paraphrase given a sentence and a target syntactic form. They evaluated their adversaries on two different sentiment analysis datasets and showed that these properly fool the model. Ribeiro, Singh and Guestrin (2018) propose semantically equivalent adversaries, i.e. semantically equivalent paraphrases, by translating and back-translating each respective sentence into another language. They generated adversarial examples for machine comprehension, sentiment analysis and visual question answering to show bugs in three SOTA models for the respective tasks. Jia and P. Liang (2017) inserted semantically correct but irrelevant paragraphs into texts to fool neural reading comprehension models. Zhao, Dua and Singh (2018) trained a Generative Adversarial Network (GAN) to generate semantically and syntactically coherent sentence adversaries, evaluated different models against it in the textual entailment and machine translation task and showed the potential of their adversaries to harm these models.

The previously mentioned attacks are mostly considered as **black-box** attack scenarios. Hereby the attacker has no access to the model parameters. Therefore the attacker can only change the inputs to the model and interpret its predictions. So no access to the architecture is given and the adversarial examples built with this approach are easier transferable to other models compared to white-box attacks. In contrast, **white-box** attacks (Ebrahimi, Rao et al., 2018; B. Liang et al., 2018; Ebrahimi, Lowd and Dou, 2018) have full access to the models' parameters, i.e., network architecture, loss functions, activation functions and weights. They can therefore calculate the most destructive change to an input for a specific model. There is also a combination of both attacks (Gil et al., 2019) where a white-box model is used to generate adversarial examples. Afterwards these examples are used to train a DNN in order to generate such white-box examples in a black-box manner.

Attackers (Hosseini et al., 2017; Rodriguez and Rojas-Galeano, 2018), which try to change the output of a model to a specific value instead of just breaking its main task, are called **targeted** attackers. This is achieved by modifying the input so that the model is predicting the desired output. In contrast, the main target of **untargeted** attacks (Pruthi, Dhingra and Lipton, 2019; Belinkov and Bisk, 2017; Alzantot et al., 2018; D. Jin et al., 2019; Eger et al., 2019) is to disturb the model in its main task performance.

An overview of current adversarial attackers is also given by W. E. Zhang et al. (2019). This paper will focus on low-level, untargeted, black-box attacks.

2.2. Robustness

Adversarial training is a commonly used technique to address adversarial attacks. Szegedy et al. (2014) have shown that inserting adversarial examples into the training data makes them somewhat robust to them. Goodfellow, Shlens and Szegedy (2014) have followed up this approach and introduced the term adversarial training. Szegedy et al. (2014); Goodfellow, Shlens and Szegedy (2014); Ebrahimi, Rao et al. (2018); Ebrahimi, Lowd and Dou (2018) use the best white-box adversary, i.e. adversarial examples are calculated in respect to the models' parameters, for each time-step in the training. In contrast, adversarial training based on black-box adversaries trains on model unspecific adversaries (Eger et al., 2019; Alzantot

et al., 2018; Belinkov and Bisk, 2017). Different manners of adversarial training can be used. Belinkov and Bisk (2017) trained models on datasets which are perturbed by one attacker and found good performances against the same kind of perturbation in the respective attack scenario. They also trained on datasets which are perturbed by multiple attackers and found performance increases against all attackers included into the training data. Their best performing model over all attackers in average was the model which was trained on a mixture of all perturbers.

Another approach is to use **spell checking** in different manners previous to the downstream task. Pruthi, Dhingra and Lipton (2019) introduced robust word recognition. They placed a word recognition model, i.e. word-based Recurrent Neutral Network (RNN) in front of the downstream classifier which predicts the correct word for every input it gets. Rodriguez and Rojas-Galeano (2018) placed a deobfuscation- and neutralization filter in front of the downstream model.

Another approach is to use different **informed embeddings** for the models. In order to prevent visual character replacement attacks Eger et al. (2019) introduced visually informed embeddings. These embeddings encode additional information about the appearance of the letters to encode similar looking letters as similar embeddings. To address attacks by replacements of synonyms X. Wang, H. Jin and He (2019) introduced the Synonym Encoding Method (SEM) whereat the model learns the same embedding for all synonyms of a word. Belinkov and Bisk (2017) used structure invariant representations i.e the average character embedding as word representation to overcome character swaps or shuffles. Additionally Belinkov and Bisk (2017) discovered that natural noise mostly consists of phonetic errors and omissions of unstressed letters. A novel approach would be to use phonetic word embeddings which encode their pronunciation.

Goodfellow, Shlens and Szegedy (2014) define that a model can be made more robust by switching to **nonlinear model** families such as radial basis function networks. They state that at the same time the model would get harder to train.

2.3. Embeddings

Both visual character embeddings and phonetic word embeddings are needed to properly replace similar looking or sounding ones by each other.

In order to encode the visual properties of letters two major approaches have been made so far. F. Liu et al. (2017), Dai and Cai (2018) and Shimada, Kotani and Iyatomi (2016) used a Convolutional Neural Network (CNN) with multiple convolution steps in which an image of a character is inserted. They reduced the dimension of the inserted image by applying multiple convolutional steps until the desired embedding dimension is reached. Eger et al. (2019) stacked the rows of the 24x24 grayscale character image representation matrix to get a 24*24=576 dimensional visual character embedding vector.

In the domain of NLP no work towards phonetic word embeddings has been considered so far. The main reason therefore seems to be the low usability in the majority of tasks. In order to attack with or defend against phonetic attacks phonetic embeddings are needed.

3. Experimental Setup

Before evaluating and reporting the performances of the neural network in chapter 4 this chapter gives an introduction to the experimental setup. Section 3.1 introduces the deep learning model *RoBERTa*, the evaluation tasks and the data we used to conduct our experiments. Section 3.2 gives a brief introduction to the methods being used to obtain the embeddings needed for the phonetic and visual attack. Section 3.3 introduces the attacks the model is evaluated on and their applications, and section 3.4 presents the approaches being used to increase the robustness of the model.

3.1. Tasks and Data

Our base architecture used in all experiments is the SOTA transformer model *RoBERTa* (Y. Liu et al., 2019). It is pretrained on a Masked Language Modeling task whereat tokens in the input get masked and it has to recover the true input. Thereby it learns contextualized word embeddings which can later be used by downstream tasks (e.g. sentiment analysis). High performaces has been shown on a variety of downstream benchmarks: GLUE (A. Wang et al., 2019), RACE (Lai et al., 2017) and SQuAD (Rajpurkar, J. Zhang et al., 2016; Rajpurkar, Jia and P. Liang, 2018). To study the performance of *RoBERTa* in an attack scenario we evaluate it on three different tasks with their respective datasets illustrated in Table 3.1.

POS tagging is a word level task where the model has to label each token in the input with its respective POS tag (e.g. *ADJ*, *NOUN*, *VERB*). The universal dependencies dataset used in this task consists of 17 different tags. The best performance reached on the full English portion of the dataset is 95.59% accuracy score (Cui and Y. Zhang, 2019).

NLI is a sentence level task in which the model has to predict the relation for a pair of sentences. Possible labels are: *neutral*, *contradiction* and *entailment*. The Stanford Natural Language Inference (SNLI) dataset is used for this task. The current SOTA performance is by Z. Zhang et al. (2019) with 91.9% accuracy score[1].

Toxic Comment Classification is a multi-label sentence classification task. The model has to label a sentence with one or multiple labels. These labels are: *toxic*, *obscene*, *threat*, *insult* and *identity hate*. For this task we choose the jigsaw toxic comment challenge dataset from kaggle[2]. The current best performance on the leaderboard has an AUCROC score of 98.8[3]. The AUCROC measures how much the model is capable

[1] *Stanford Natural Language Inference Website* (2020). URL: https://nlp.stanford.edu/projects/snli/ (visited on 22/02/2020).

[2] ConversionAI-Jigsaw (2020). *Kaggle Toxic Comment Classification Challenge.* URL: https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge (visited on 20/01/2020).

[3] *Jigsaw Toxic Comment Classification Leaderboard* (2020). URL: https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/leaderboard (visited on 20/01/2020).

of distinguishing between the different classes. In addition this task is a good application scenario because it has practical relevance (e.g. spam filters, hate speech detection).

Task	Dataset	Train	Test	Best score from literature
POS Tagging	Universal Dependencies (part)	13k	2k	95.59%
NLI	SNLI	550k	10k	91.9%
Multilabel-Classification	Toxic Comment	560k	234k	0.98

Table 3.1.: Overview of the NLP tasks used in this work. Additionally the data splits and the SOTA performance are given. These are measured in accuracy for POS and NLI, and in AUCROC score for multilabel-classification.

3.2. Embeddings

This section will give a brief introduction to the methods being used to generate the character- and word-based embeddings for the phonetic and visual attack.

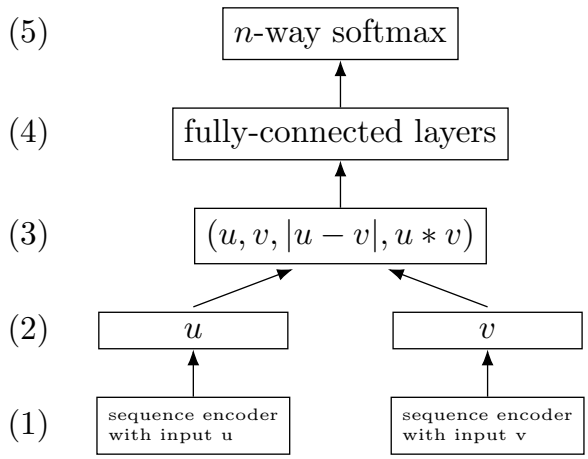
3.2.1. Phonetic

In order to replace words by their phonetic neighbors we build phonetic word embeddings. We therefore adapted the architecture of *InferSent* first introduced by Conneau et al. (2018). The following introduces the main points of the original architecture of *InferSent* and explains the adapted version *WordSim*.

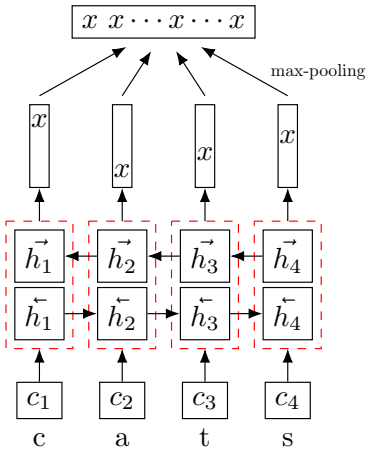
WordSim

InferSent has been adapted to compute phonetic word embeddings. The architecture is illustrated in Figure 3.1a. We replaced the sentence encoders by character-based word encoders. The word encoder (1) is based on the best performing encoder architecture of *InferSent*: BiLSTM max-pooling shown in Figure 3.1b. The Bi-directional LSTM (BiLSTM) max-pooling encoder consists of two Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) units. They read the input sentences in two opposite directions. Irrespective of the reading direction each LSTM reads a sequence of N words $w_1, ..., w_N$ and computes a set of N hidden representations $h_1, \ldots, h_N$. Each hidden state is calculated based on the current word and the last hidden state: $h_n = LSTM(w_1, ..., w_n)$. In a unidirectional manner the sentence is therefore represented by the last hidden vector h_N. The forward LSTMs' hidden states are represented by $\vec{h}_n$ and the backward LSTMs' hidden states are represented by $\overleftarrow{h}_n$ where n is the n'th word. To get the sentence representation in the bidirectional approach the hidden states of both LSTMs are concatenated and max-pooling is applied. As a result both words are individually embedded into an 100-dimensional word embedding: u, v (2). In contrast to *InferSent* both embedding layers share their parameters, because it does not matter in which input we enter which word. Once the word vectors are computed, two additional features based on these vectors are calculated to extract relations between them: the element-wise product $u \circ v$ and the absolute element-wise difference $|u - v|$. We concatenate (3) the two word vectors, the product and the difference to a vector which is fed into the fully-connected Multilayer Perceptron (MLP)

(4) with a softmax layer (5) at the end. This softmax distinguishes between 4 classes of phonetic similarity between the input words: *identical*, *very similar*, *similar* and *different*.



(a) Abstract architecture of *InferSent* ($n = 3$; u,v: sentences) and *WordSim* ($n = 4$; u,v: words)



(b) BiLSTM encoder architecture.

Figure 3.1.: Images were taken from (Conneau et al., 2018) and adapted to match *WordSim* architecture.

We built our own dataset for phonetic similarity by crawling data from different sources, because no such data was available yet. A good starting point were pronunciation dictionaries like *Combilex* (Fitt, Richmond and Clark, 2020) which assign each word its associated phoneme. To obtain a comparable measurement of phonetic similarity, we calculated the edit-distance between the phonemes of each word pair and divided it by the length of the shorter phoneme to normalize it:

$$sim_{ph}(p_1, p_2) = \frac{d(p_1, p_2)}{min(|p_1|, |p_2|)} \tag{3.1}$$

where $p_i \in \{1, 2\}$ are the phonemes and d is the edit-distance. The resulting value represents the phonetic similarity percentage between the two input words. Afterwards these words were clustered into 4 different classes: *identical* ($sim_{ph} = 0$), *very similar* ($0 < sim_{ph} < 0.1$), *similar* ($0.1 < sim_{ph} < 0.3$) and *different* ($0.3 < sim_{ph}$) to build our training bins. The resulting class distribution is shown in Figure 3.2. As expected there are few samples in *very similar* and a lot of samples in the *different* bin. To keep the training data for each class balanced we added handcrafted and crawled samples by hand, e.g. homophones, [4]. There was

[4] *Homonym: List of 300+ Homonyms in English with Examples* (2019). URL: https://7esl.com/homonyms/ (visited on 12/12/2019)
EnglishClub.com (2019). *Homophones List*. URL: https://www.englishclub.com/pronunciation/homophones-list.html (visited on 12/12/2019)
Homophones List (2019). URL: http://homophonelist.com/homophones-list/ (visited on 13/12/2019)
Ian Miller (2019). *Homophones*. URL: http://www.singularis.ltd.uk/bifroest/misc/homophones-list.html (visited on 12/12/2019)
Multinyms Triple/Quadruple/Quintuple/Sextuple Homonyms (30th Oct. 2015). URL: https://web.archive.org/web/20160825095711/http://people.sc.fsu.edu/~jburkardt/fun/wordplay/multinyms.html (visited on 12/12/2019)
Richard Nordquist (16th July 2019). *200 Homonyms, Homophones, and Homographs*. URL: https://www.thoughtco.com/homonyms-homophones-and-homographs-a-b-1692660 (visited on 13/12/2019)
Appendix:English dialect-independent homophones (2019). URL: https://en.wiktionary.org/wiki/Appendix:English_dialect-independent_homophones (visited on 11/12/2019)
Appendix:English dialect-dependent homophones (2019). URL: https://en.wiktionary.org/wiki/Appendix:English_dialect-dependent_homophones (visited on 11/12/2019)

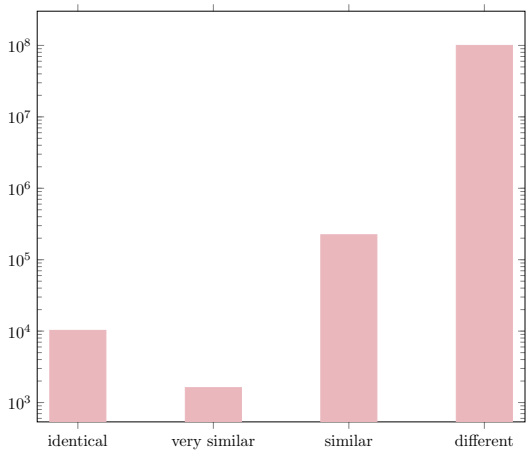


Figure 3.2.: Number of samples per similarity class calculated for the Combilex pronunciation dictionary.

Homophone	byte	bite
Abbreviation	I love you too!	I luv U 2!

Table 3.2.: Example of a homophone and a typical ”internet slang” abbreviation.

also a lack of “internet slang” style phonetic replacements like in Table 3.2. We therefore crawled[5] them manually and added them to the bins *identical* and *very similar* with best knowledge and certainty. After adding new data to the emptier bins there were finally enough data to slice it into four equal bins with 5k samples each. The *identical, similar* and *different* bins consist only of data from Combilex, whereas the *very similar* bin contains of 1.3k samples from Combilex and 3.7k crawled samples.

3.2.2. Visual

In order to generate character embeddings we used an architecture introduced by Larsen et al. (2016) as a combination of GAN and Variational Autoencoder (VAE). A schematic overview of a Variational Autoencoder Generative Adversarial Network (VAEGAN) is given in Figure 3.3. The model is able to learn embeddings which encode high-level abstract features, i.e. wearing glasses, by optimizing a not pixel-based loss function. This property is desirable in our case, because humans rely on abstract features (Dehaene and Cohen, 2011), i.e. shape and spatial relation of the letter, instead of pixels while reading. The following will give a short introduction to the main aspects of the architecture before we move on to our setup to compute the visual embeddings.

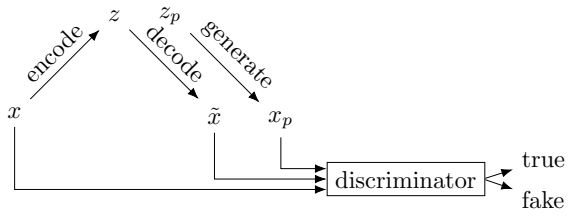
The model reduces the dimension of input x, e.g. an image, by applying multiple convolutional steps in the *encoder* to compute the latent representation of x: z. Afterwards it reconstructs the original input x in the *decoder* by applying multiple deconvolutional steps to z. This reconstructed version of x is called $\tilde{x}$. Additionally z_p a second input sampled from $\mathcal{N}(0, I)$ is inserted into the *generator* to obtain x_p. Although

[5]Heiko Possle (2019). *Text Message — SMS — E-Mail — Chat*. URL: https://www.smart-words.org/abbreviations/text.html (visited on 10/12/2019)
Vangie Beal (2019). *Huge List of Texting and Online Chat Abbreviations*. URL: https://www.webopedia.com/quick_ref/textmessageabbreviations.asp (visited on 11/12/2019)

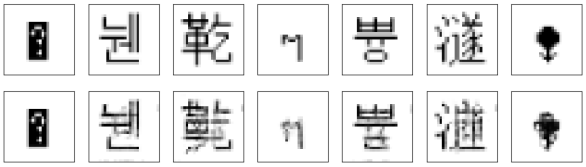
the *decoder* and the *generator* execute the same task, they are considered as identical and therefore share their parameters. The last remaining component is the *discriminator*. It takes $x, \tilde{x}$ and x_p as inputs and discriminates which input is a real training sample and which is a fake. The peculiarity of generative models like this VAEGAN is that they improve the performance of their components by having them compete against each other. While the generator is improving its performance of generating samples, which match the original data distribution, out of noise, the discriminator gets better at distinguishing between real and fake images. In addition a specific property of the VAEGAN is that it also improves the ability of the encoder/decoder to properly compress/decompress the samples/latent representations in this process.

Finally to obtain our visual character embeddings we generate a grayscale image of size 24×24 for each character in the Basic Multilingual Plane (BMP) of the Unicode standard with *Pillow*[6]. Thereby we got 65k different images of characters later used as input x for the VAEGAN. In our case the *encoder* applies multiple convolutional and max-pooling steps to the image to compress it. The *decoder* and *generator* apply multiple deconvolutional and sampling steps respectively. The model is trained on the full BMP dataset. Afterwards we computed our 256-dimensional visual letter embeddings by encoding the respective letter image with the encoder of the VAEGAN. The models' capability to properly reconstruct an image from its embedding is shown in Figure 3.3. The better the model has encoded the visual features of the image the more identical the two pictures are.

Figure A.1 gives an impression of the encoded visual similarity.



(a) Schematic representation of VAEGAN taken and adapted from Larsen et al. (2016). z can be decomposed as $z = \mu + \sigma$ and is used to sample $z_p = \mu + \sigma\epsilon$ where ϵ is noise defined as $\epsilon \sim \mathcal{N}(0, I)$



(b) Reconstruction of images after being compressed to its latent representation and decompressed back to the original data distribution.

Figure 3.3.: VAEGAN architecture and reconstruction.

3.3. Attacks

This section introduces the attacks used to harm SOTA models like *RoBERTa*. Examples for each kind of perturbation can be found in Table 3.3. We introduce three different attack levels with perturbation probability p: *low* ($p = 0.2$), *mid* ($p = 0.5$) and *high* ($p = 0.8$) to evaluate the model against different strengths of perturbations. Their main difference is the probability to apply a perturbation to a datasample. In order to perturb the datasets we apply the following process to each sample x:

The word (token) sequence of each sample can be represented as $x = [w_1, ..., w_n]$. We calculate the number of tokens to be perturbed t_{target} by multiplying the token count $|x|$ with the perturbation probability p.

[6]Fredrik Lundh, Clark Alex and Contributors (2020). *pillow - the friendly PIL fork*. URL: https://python-pillow.org/ (visited on 01/01/2020).

Afterwards we pick a random token w_i and perturb it if possible (e.g. if a phonetic perturbation of the token is existent). We repeat this process until we perturbed t_{target} tokens or every token in the input has been considered.

We perturb each downstream task dataset introduced in section 3.1 with the three different attack levels and one perturber. We repeat this for every perturbation listed in the following and therefore get 30 different datasets for each task.

3.3.1. Perturbations

Inner Shuffle. The first kind of perturbation is the randomization of all letters in a word except the first and last. This attacks utilizes the ability of humans to still comprehend words if the first and last letter remain untouched (Rayner et al., 2006). Due to the omitting of the first and last letter in the randomization process, change would only occur in words with length ≥ 3. Therefore they are skipped.

Full Shuffle. This is the extreme case of the inner-shuffle perturbation. Here the positions of all letters of a word are randomly exchanged. We assume that the restoration of short words should not be difficult for experienced readers and because the letters of words of length 1 cannot be exchanged this attack applies to all words with length ≥ 2.

Intruders. The third attacker inserts different kinds of punctuation into the word. The inserted symbol is chosen randomly but in case of multiple insertions into one word it remains the same. Possible symbols are: !"#$%&'() $*+,-./:;<=>?@[\]^_`{|}~. The perturbation probability additionally influences the number of insertions taking place. For each two characters p indicates how likely the insertion of a symbol between them is. Because no symbol can be inserted between words with length 1, this attack only applies to words with length ≥ 3.

Disemvoweling. This data perturbation will remove all vowels (i.e. a, e, i, o, u) from a word. If a word only consists of vowels it will be ignored to prevent it from being deleted. Words with length ≤ 3 are skipped to maintain the readability.

Truncating. Remove a fixed number of letters, e.g. 1, from the back of a word to a minimum word length of 3. It does not apply to words of length ≤ 3 to maintain the readability.

Segmentation. Multiple words are joined together into one large word. In contrast to the other attacks this perturbation is applied differently. Hereby the perturbation level is the probability to merge the first two adjacent words. Each following word gets a lower probability to get merged $(p^2, ..., p^n)$. The probability to join the nth word w_n equates to $p(w_n) = p^n$ where p is the perturbation probability. This attack cannot apply to the POS tagging task, because the segmented words have no proper tag.

Keyboard Typos. This attack has been taken from Belinkov and Bisk (2017) and has been adapted to our workflow. Hereby adjacent letters on the English keyboard are replaced by each other randomly. This simulates human typing errors. The higher the perturbation probability p the more characters are exchanged by adjacent letters.

Natural Typos. This attack has also been taken from Belinkov and Bisk (2017). Words are replaced by natural human errors from the Wikipedia edit history. This dataset[7] contains multiple sources of error: phonetic errors, omissions, morphological errors, key-swap errors and combinations of them.

[7] Yonatan Belinkov and Yonatan Bisk (2020). *Github Repository containing scripts and noise data for (Belinkov and Bisk, 2017)*. URL: https://github.com/ybisk/charNMT-noise (visited on 27/02/2020).

Phonetic The phonetic attack is based on a sequence-to-sequence architecture which consists of two models. The first model performs a grapheme-to-phoneme (G2P) conversion to predict the phonemes for a given sequence of graphemes. The second model performs the inverse phoneme-to-grapheme (P2G) conversion. In combination the G2P-P2G model translates a sequence of graphemes into its respective phonemes and back. Both models have been trained independently on two disjunct parts of the *Combilex* (Fitt, Richmond and Clark, 2020) dataset for their respective tasks. The full dataset consists of 100k word-pairs. The better they perform on their tasks the more similar the original and the double-translated word are. A desirable result would be a small difference in the graphemes whilst keeping the phonetics identical. Additionally we introduce a constraint based on the phonetic word embeddings introduced in section 3.2.1 to make sure they are similar. Therefore we calculate the similarity between the original input and the double translated output and only apply the perturbation if *WordSim* classifies them as *identical* or *very similar*.

Visual The visual attacker is based on *VIPER* (Eger et al., 2019) and operates on the full input sentence x. It takes the probability p and an embedding space, i.e. image-based character embedding space (ICES) computed in section 3.2.2, as arguments. For each character in the input sequence a perturbation decision is made with respect to p. If a replacement should take place the letter is replaced by one of its 20 nearest neighbors with respect to the ICES.

Attack	Mode	Perturbed Sentence
None	-	Adversarial attacks are harmless to all NLP models.
Inner Shuffle	low mid high	Adversarial akcatts are hlrmases to all NLP models. Aadrreavsil aacttks are hmarsels to all NLP mdeols. Aaivrasredl aactkts are hrsaemls to all NLP moleds.
Full Shuffle	low mid high	Adversarial katctas are harmless to all LNP models. idaAasvrler tstkaac are harmless to lla LPN elmdos. arsiAvleadr kastact rae raslshem to lla PNL oslemd.
Intrude	low mid high	Adversarial attacks are harmles‘s to all NLP m-od-e-ls. A d v e r sar ial at:ta:ck:s are h}ar}m}less t$o a(ll NLP models. A/d/v/e/r/s/a/rial at?t?a?c?k?s a:re h*a*r*mle*s*s t_o al.l N'L'P m'o'd'e'l's.
Disemvowel	low mid high	dvrsrl attacks are hrmlss to all NLP models. dvrsrl ttcks r hrmlss to ll NLP models. dvrsrl ttcks r hrmlss to ll NLP mdls.
Truncate	low mid high	Adversarial attack are harmles to all NLP models. Adversaria attack are harmles to all NLP model. Adversaria attack are harmles to all NLP model.
Segment	low mid high	Adversarial attacks are harmlessto all NLP models. Adversarial attacksare harmless to allNLPmodels. Adversarialattacksareharmless toallNLPmodels.
Typo	low mid high	Adversarial attacks are harmless to akl NLP ,odels. Adverssrial attaxks are harmless yo al. NLP ,odels. Axversarial attafks arf harmpess tk alo NLP nodels.
Natural Noise	low mid high	Adversarial attacs are harmless to all NLP modeles. Adversarial attacs rae harmless tio alle NLP modeles. Adversarial atacks arre harmless t0 All NLP modeles.
Phonetic	low mid high	Adveraciory attacks are harmless to all NLP moddles. Adveraciory attacks are harmless to all NLP models. Adveraciory attacks are harmless two all NLP moddles.
Visual	low mid high	Ädveʀṣaȓial åttacks arɜ harṃless to ałl NLP mȭdels. Ǟḋ6ɝrsarīał aṯtacḳs āɿe hǟrʷʟḕṩ to ǎl NⱣ mođěls. Äɖvęʳãřiaˡ ąẗтaḈķß åŕɜ hrɰɭɝßs tō åḻḶ NƬP moɗěls.

Table 3.3.: Simple attacks

3.4. Defenses

We evaluate three forms of shielding against our proposed attackers. Adversarial training as standard approach to increase the robustness, because its functionality has been proven by the literature, and two new developed methods: perturbed language model and perturbed pretraining. To increase the

performance against all attack levels (*low*, *mid*, *high*) at once, we insert an equal portion of each level into the training data. We call this the low-mid-high (LMH) dataset. The words that remained untouched (low: 80%, mid: 50%, high: 20%) in the perturbation datasets mentioned in section 3.3 add up to about $(80\% + 50\% + 20\%)/300\% = 50\%$ in total. Therefore half of the LMH dataset consists of perturbed data and the other half of clean samples.

3.4.1. Adversarial Training

We refer to adversarial training as inserting perturbed data into the training process (Belinkov and Bisk, 2017) to make the model more robust against the specific type of attack. We train and evaluate our model with two different strategies to measure the performance of *RoBERTa* and reveal common ground between the individual perturbations:

1-1 In this setup we finetune *RoBERTa* individually for each of the three downstream tasks on the LMH dataset of one attacker. Hereby it learns to accomplish the task on perturbed data of one kind.

Leave-One-Out In this setup the model is finetuned on a mixture of LMH datasets of every attacker except one. The remaining perturber is used afterwards to attack the model and evaluate its performance. We call it a leave-one-out fashion. In order to find out how high the impact of adversarial training on a wide variety of perturbed data is to different non-seen attackers. This case is particularly realistic, because it is nearly impossible to train a deep learning model on every existing noise. To obtain the train dataset we split each perturbers' LMH dataset into n equal parts where n is the number of different attackers being used in the training process. We take one split of each LMH dataset to form the training dataset. The resulting dataset contains the same amount of data as the original one and no example occurs twice. Thereby no discrepancy between the clean and the perturbed dataset arise.

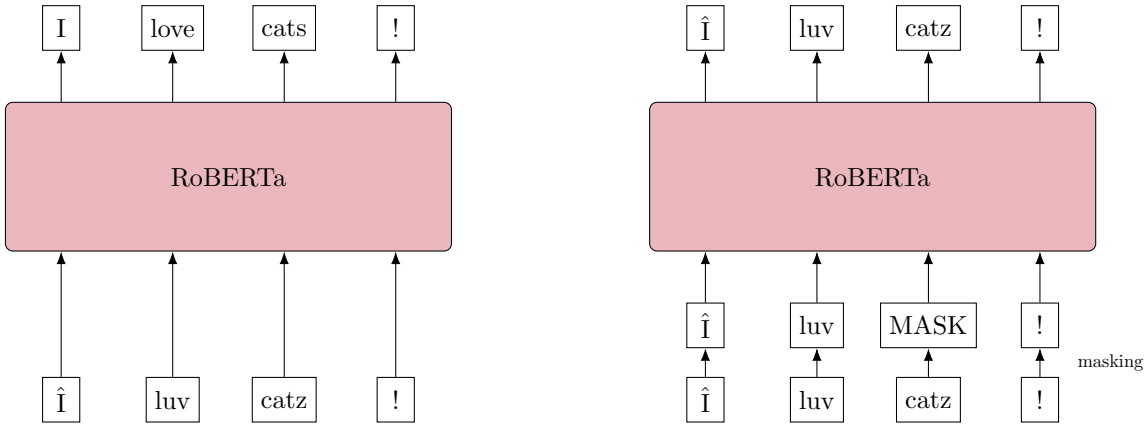
3.4.2. Perturbed Language Model

The Perturbed Language Model (PLM) task is a variation of the MLM task used to pretrain *RoBERTa* and shown in Figure 3.4a. Thereby input tokens are replaced by a *MASK* token and the model has to recover the underlying word. Instead of masking input tokens we use a perturbed version of the sentence as input to the model and the models' task is to recover the original unperturbed version of the sentence. Hereby we want the model to learn to unperturb an input example to its original version. We assume that this will improve its performance against perturbed inputs for the individual downstream tasks. We ran PLM for 300 epochs on a portion (40k sentences) of the wiki-103 dataset (Merity et al., 2016) perturbed by all attackers in the LMH favor. Afterwards we finetuned it on clean data for the individual downstream tasks POS, NLI and TC and evaluated it against all available attackers.

3.4.3. Perturbed Pretraining

The original MLM task used to train *RoBERTa* (Y. Liu et al., 2019) was executed on a large mostly clean corpus. This is because perturbed input was not intended to be taken into account in the pretraining process. In order to familiarize the model with perturbed inputs we continue the MLM task on perturbed data. The same perturbed portion of the wiki-103 dataset mentioned in 3.4.2 is used. So while training the model random words in the input are replaced by the *MASK* token and the model has to predict the

original input of the masked word based on its context. This process is visualized in Figure 3.4a hereby we want the model to learn representations for the perturbed words. In the best case the embeddings of clean and perturbed versions of the words are somewhat similar and thereby increase the robustness of the model to such perturbed inputs. We pretrain the model for 300 epochs and afterwards finetune it on clean data individually for all downstream tasks.



(a) Perturbed Language Model

(b) Perturbed Pretraining

Figure 3.4.: Perturbed Language Model and Perturbed Pretraining

4. Results

This chapter will present the results of our experiments which will be analysed and discused in chapter 5. In section 4.1 the results of the attack experiments are reported. In section 4.2 the robustness changes of the three different shielding approaches are reported.

4.1. Attacks

In Figure 4.1 we plot the performance of *RoBERTa* for the three tasks POS, NLI and TC individually as we perturb the test data using our attackers. For detailed data consult Table A.2 in the appendix. The x-axis represents the different attack levels, the y-axis reports the relative performance equal as in Eger et al. (2019)

$$s^*(p) = \frac{s(p)}{s(none)}, \quad p \in \{none, low, mid, high\} \tag{4.1}$$

where $s(none)$ is the task specific performance on clean data listed in Table 4.1 and $s(p)$ is the performance under attack with attack level p. $s^*(p)$ is measured in accuracy for POS and NLI, and in AUCROC for TC classification. An important note about the performance measurements is that they depend on their respective task and dataset. That means the NLI task reaches its worst performance at around a score of 33% accuracy, because it assigns the majority class label to each input, 16% accuracy for POS respectively. Additionally the worst performance of TC is reached on 0.5 AUCROC score - at this point the model is no longer able to distinguish between the different classes. We mark this values relative to the tasks best performance in Figure 4.1 as red line defined as:

$$s_t^{worst} = \frac{w_t}{s(none)} \tag{4.2}$$

where w_t is the above mentioned worst value that can be reached in task t.

We can see performance decreases for every attacker in every task. Overall the higher the perturbation level, the lower the models' performances. In the following the respective task-specific model is referred to as 'the model'.

The **phonetic** attack is the least effective for all tasks with 10% performance decrease as its best with the *high*est perturbation probability. Additionally the three different attack levels almost do not differ.

	POS	**NLI**	**TC**
$s(none)$	96.65	90.41	0.93

Table 4.1.: Performance on clean test data for the respective downstream tasks.

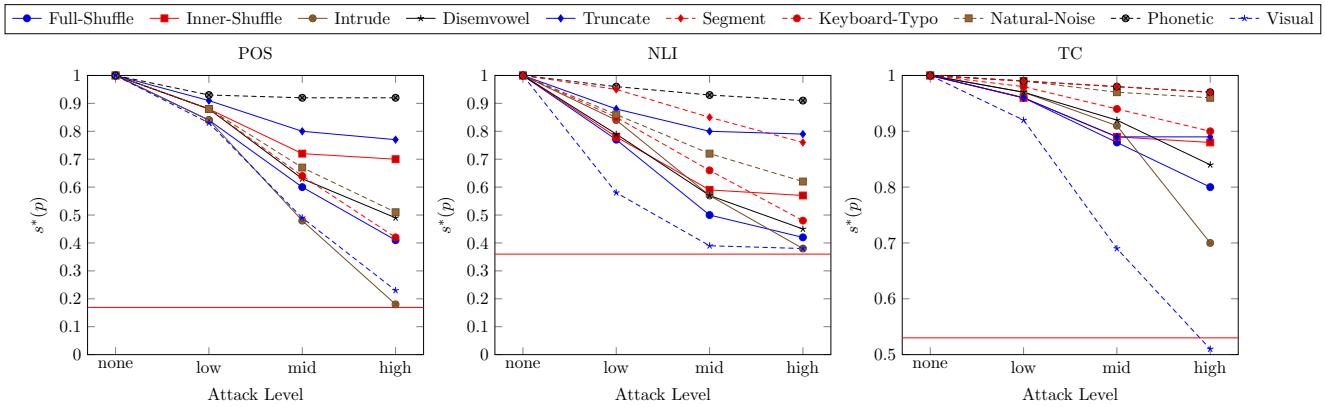


Figure 4.1.: Performance of undefended model against all adversarial attacks introduced in 3.3 for POS left, NLI mid and TC right. Performance measured in $s^*(p)$ defined in Equation 4.1. The red lines indicate the worst performance that can be reached relative to the models' best performance for the respective task and normalized by $s(none)$.

The **truncate** attack yields better results as the phonetic attack in all three tasks. It is therefore somewhat twice as effective. The performance decreases by 10% from *none* to *low* and from *low* to *mid*. Furthermore the increasing attack level from *mid* to *high* has almost no impact on the performance on the downstream task. Especially the sentence level tasks do not suffer noticeably more by increasing the perturbation level to *high*.

In addition the sentence based tasks were evaluated against the **segmentation** attack. For NLI it accomplishes a similar performance decrease for the model as the truncate attack. In contrast it gets linearly more successful as the perturbation level increases from *low* to *mid* and *high*. For TC the performance is almost identical to the phonetic attack. NLI suffers a lot more than TC by around 25% rather than 5%.

We notice a linear decrease in performance for each task while increasing the perturbation level of the **natural-noise** attack. In TC it has a similar low impact to the model as the segmenation attack by around 5% for *high*. This is in contrast to POS and NLI which suffer a decent performance deterioration of around 40% and 50% for the *high*est attack level. Both lose 15% to 20% performance per attack level raise.

Full- and **inner-shuffle** are applying roughly the same kind of perturbation to the data. In general all tasks are suffering from both methods. In the POS and TC task the performance of the model decreases linearly as the full-shuffle attack level increases to a maximum at probability *high* where it degrades by 40% and 20% relative performance respectively. For NLI the performance deficit is around 25% each for the increment from *none* to *low* and *low* to *mid* and lowers to 10% for the transition to *high* to a maximum of 60%. In contrast the inner-shuffle reduces the models' performance linearly while increasing the level from *none* to *low* and *mid* for all tasks. In detail by 15% for POS, 20% for NLI and 5% for TC. Increasing the level to *high* seems to have a low impact.

The **disemvowel** attack has different effects on the models' performance in the different tasks. For POS it nearly aligns to the natural-noise attack with a slightly better performance of 5% for *mid* and 3% for *high* and a maximum on 50%. NLI loses around 20% performance on *low* and it decreases an additional 20% by increasing the level to *mid*, while reaching its greatest decrease by 55%. In TC the models' performance decreases linearly from *none* to *low* and *mid* by 8% each. The *high* attack level doubles this decrease to a total of 15% performance loss.

The **intrude** attack is one of the best performing attacks for all three tasks. On TC the *low* and *mid* attack levels have a relatively low impact compared to *high* which yields a performance loss of 30%. In addition it decreases the models' performance the most on the POS task by more than 80%. It still performs marginally better for the POS task.

The **keyboard-typo** attacks linearly decreases the models performance with respect to the attack level. NLI and POS start on around 15% total decrease on *low* and decrease further to 35% for *mid* and 55% for *high*. Therefore this attack has a high impact on the model. In contrast the performance decreases for the TC task are lower compared to the other two tasks. It only drops to a maximum of 10% at the *high*est attack level.

Especially for the both sentence-based tasks NLI and TC the **visual** attack decreases the model nearly to its worst possible performance. The model performs marginally better for the POS task. Important to note is that even for the *low* perturbation level the NLI model suffers from more than 40% performance decrease. It maximizes with perturbation level *mid* on 60% performance decrease. For TC the *low* attack level only impacts the models' performance by 10%. This increases to 30% for *mid* and 50% for *high*. The performance for high even goes beyond the red line marked as the worst performance possible. This is possible because the AUCROC is defined in the interval $[0, 1]$. Values below 0.5 indicate that the model confuses the labels in a way it predicts 0 for 1 and vice versa.

4.2. Defenses

In the following figures the performances of the three different tasks are reported in the same way as already mentioned above. We replace the value on the y-axis by the performance of our methods to make the model more robust as relative difference equal to Eger et al. (2019) and defined as:

$$\Delta_\tau := \frac{\sigma(p)}{s(none)} - s^*(p) \tag{4.3}$$

where $\sigma(p)$ is the score for each task with one of the defense methods τ being introduced in section 3.4. Therefore a score of 0.3 indicate 30% performance (robustness) improvement compared to the non-shielded model for a respective task, attacker and level. It is important to mention that an improvement in the following do not take place from one attack level to another but relative compared to the performance of the non-shielded model against the respective attacker. We describe the performance differences of each shielding approach individually.

4.2.1. Adversarial Training

1-1 In Figure 4.2 we report the performance of our models each trained on perturbed data and evaluated against the same kind of perturbation. The adversarial trained model on the respective task is mentioned in the following as the model.

The first thing to notice for POS is that the adversarial trained models lose a little bit of their performance on clean data and that overall the models' performance against the proposed attackers gets better. On *low* the performances against intrude and truncate have to be particularly mentioned. Against the truncate attacker the robustness of the model increases by 15% and for intrude by 10%. The performances for the remaining attackers are very similar and range from 3% increase for the natural-noise attack to 8% for

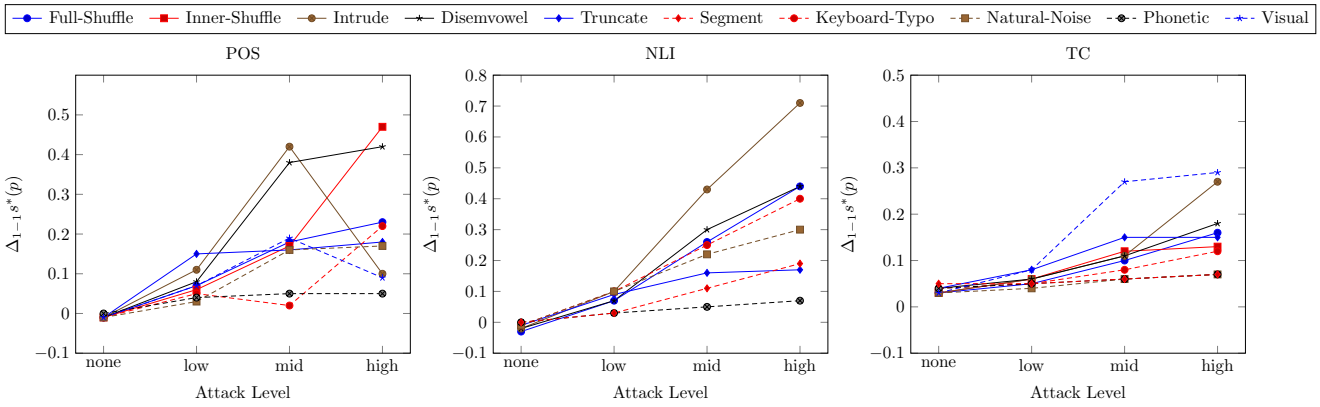


Figure 4.2.: Performance improvements of the models adversarial trained and evaluated individually on the attacker introduced in 3.3 for POS left, NLI mid and TC right. Performance measured in $\Delta_\tau(p)$ defined in equation 4.3.

the disemvowel attack. The *mid* attack level further differentiates. A rapid climb is accomplished against the intrude and the disemvowel attack with a total performance increase of 42% and 38% respectively. The performance against truncate and phonetic mostly remains unchanged. The performances against the remaining four attackers increases to a total performance improvement of around 18%. For *high* the performance against inner-shuffle increases to 47%. Against the keyboard-typo the models' performance increases to 22%. The performance against phonetic, natural-noise, truncate and full-shuffle remains roughly unchanged or lift kindly, e.g. by 5% for full-shuffle compared to the performance improvements against attack level *mid*. In contrast, the performance against intrude and visual drops to around 10% total improvement.

An important note to take first for NLI is that both the visual and the inner-shuffle models' performances are missing from the data. This is because, we were unable to train the models to perform the NLI task on the perturbed dataset. Except for phonetic and segment the models' performances drop by to -3% on clean data. Additionally all performances increase strictly as the attack level increases. For *low* the total performances increases stay close together. Against phonetic and segment the performance increases remains unchanged. In contrast, the remaining models increase their performance to 8% in average, e.g. natural-noise to 10%. Especially for attack level *mid* the performance against the intrude attack increases significantly to 43%. The performance against disemvowel increases to a total improvement of 30% and the performance against the natural-typo to 22% respectively. The performances against full-swap and keyboard-typo increase to around 25%. The models' performance against truncate and segment increases to 18% in average. The performance against the phonetic attack again increases nearly imperceptible by 2%.

As a specialty in the TC task the models' performances gain a performance increase on clean data of 4% in average. For attack level *low* only the performance increase of visual and truncate stand out slightly to a total of 8%. The performances against the remaining attackers increase to 5% on average. For *mid* the gap between visual and the remaining attackers enlarges significantly. The performance against the visual attack is increasing to 27%. In contrast, performances against phonetic and natural settle down below. The performances against the remaining attackers increase to 11% on average. For *high* the performance against the visual attacker flattens its increase and remains on 29% performance improvement in total. A clear performance increase is accomplished against the intrude attack to 28%. The performances against

truncate and inner-shuffle stay nearly unchanged and the remaining performances keep their behavior and increase to 15% on average.

Leave-One-Out In Figure 4.3 we report the performance of our models each trained on a mixture of all attackers except the one it is evaluated on.

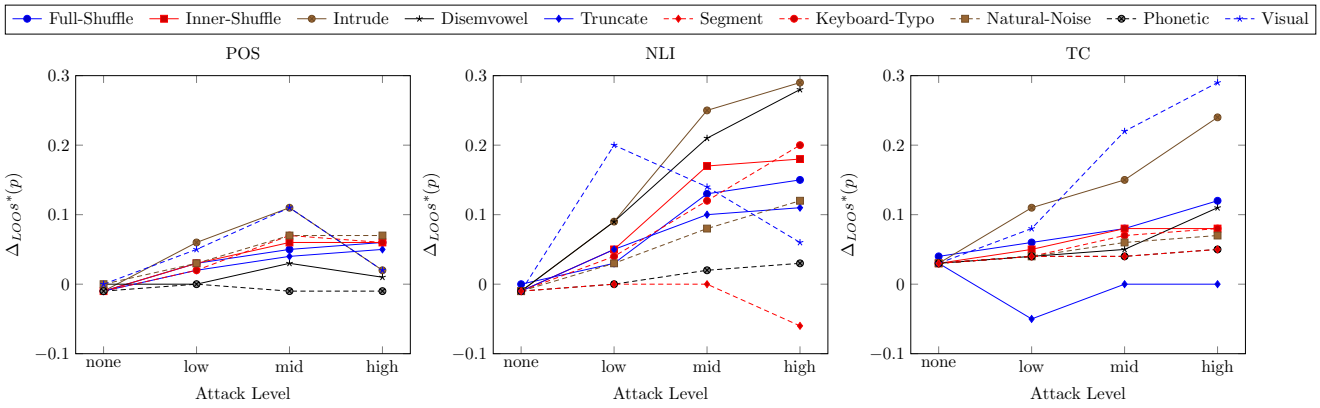


Figure 4.3.: Performance improvements of the models adversarial trained on all attackers introduced in 3.3 except the one they are evaluated on for POS left, NLI mid and TC right. Performance measured in $\Delta_\tau(p)$ defined in equation 4.3.

For POS there can be differentiated between three groups that behave similar. The performance against the phonetic attack remains mostly unchanged. The models' performance against natural-noise, inner-shuffle, full-shuffle, truncate and keyboard-typo improves with increasing attack level. The best performance of this group is accomplished against natural-noise with 3% for *low* and 7% for *mid* and *high*. The performances against the remaining three attackers visual, intrude and disemvowel are characterized by a significant lower value on attack level *high* compared to the value on *mid*. The performance against disemvowel differs from the other two by only increasing to 5% instead of 12%.

For NLI the performance against keyboard-typo, full-shuffle, inner-shuffle, natural-noise and truncate exhibits steady improvements with increasing attack level which range from 10% to 20% for attack level *mid* and *high*. The performances against intrude and disemvowel also show steady improvements with the attack levels whereas these improvements are significantly higher with up to 29%. For attack level *low* the performance against the visual attacker is with 20% more than twice the value as the others. This improvement diminishes in the *mid* and *high* attack level and even drops below the improvements against most of the other attackers. Particularly noticeable is the deterioration against segment at attack level *high* to -6%.

In the TC task the models' performance against the visual and intruder attack attract attention. The performance against visual improves even for *low* level to 8%, increases for *mid* to 23% and maximizes to 29% total improvement for attack level *high*. The performance against the intrude attack on attack level *low* indicates a higher value (11%) compared to the visual improvement (8%). However, it does not result in a performance increase as high as against visual and only increases to 15% for attack level *mid* and 25% for *high* respectively. The performances against full-shuffle, inner-shuffle, disemvowel, segment, keyboard-typo, natural-noise and phonetic behave similar for attack level *low* and *mid* with 4% to 7% total improvement. Additionally the performances against full-swap and disemvowel further increase compared

to the before mentioned to around 12% total increase. The previously inconspicuous performance against the truncate attack deteriorates by 5% for attack level *low* and remains unchanged for *mid* and *high*.

4.2.2. Perturbed Language Model

In Figure 4.4 we report the performance changes for the models pretrained on the PLM task, in which the model has to unperturb the words to their original input, compared to the clean models.

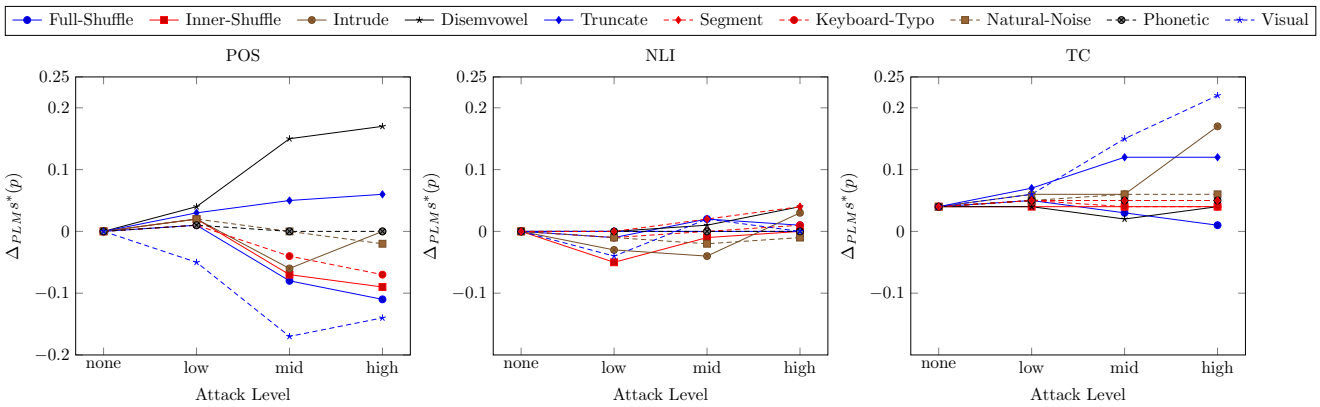


Figure 4.4.: Performance improvements of the models pretrained on the PLM task against all adversarial attacks introduced in 3.3 for POS left, NLI mid and TC right. Performance measured in $\Delta_\tau(p)$ defined in equation 4.3.

For POS the robustness increases against the individual attacks are very different. The robustness against the disemvowel attack profits strikingly strong compared to the remaining attackers. Additionally the performance against the truncate attack increases to around 5% for all attack levels. The performances against the phonetic and natural-noise attacks remain unchanged. Against the keyboard-typo, full- and inner-shuffle attack the performance decreases by an additional 5-10% scaling with the increasing attack level to a maximum of around -10% on attack level *high*. Intrude and visual are related with respect to their shapes where the models' performance drops further for the *mid* level than for *low* and *high*. For intrude the model has a performance decrease of 5% for the *mid* attack level and remains unchanged for the other levels. The robustness against the visual attacker stands out in its heavy loss of -5% for *low*, -17% for *mid* and -14% for *high*.

For NLI the performance on clean data does not suffer from applying the PLM beforehand. All robustness changes are within an interval from -5% to 5%. Only the performances against the segment and the disemvowel attack improve slightly for all attack levels to a maximum of 5%. The performances against the truncate and phonetic attack remains unchanged. The performances against the remaining attacks is influenced positive and negative depending on the attack level.

For TC, the performances against all types of perturbations initially increase by 4%. No further improvements in performance could be determined against the inner-shuffle, segmentation, phonetic and keyboard-typo attacks. The performances against the disemvowel attack remains unchanged with a small drop for attack level *mid*. Performances against full-shuffle drops with increasing attack level and is getting closer to the original performance (0%) on the clean model. The performance against natural-noise increases slightly with the raising attack level. The performances against truncation raises to 12% for attack level *mid* and

high. Against intrude the performance increases especially for attack level *high* The performance of the model against the visual attack is increasing the most to 15% for *mid* and 23% for attack level *high*.

4.2.3. Perturbed Pretraining

Figure 4.5 visualizes the performance increases and deteriorations of the model against the different attackers compared to the performances of the clean model. Every plot represents one individual model first pretrained on data perturbed by all attackers and afterwards finetuned on the respective downstream task with clean data.

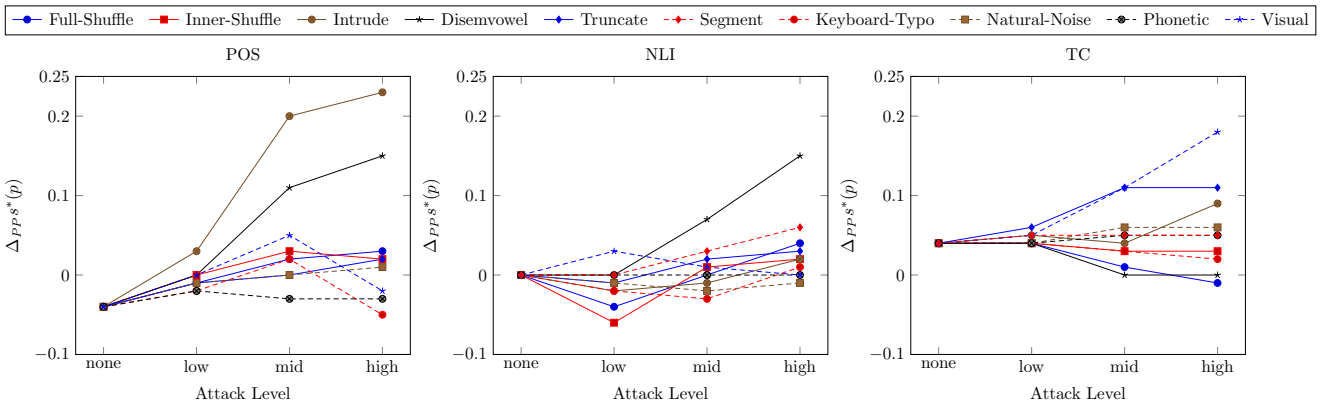


Figure 4.5.: Performance improvements of the models pretrained on perturbed data of all adversarial attacks introduced in 3.3 for POS left, NLI mid and TC right. Performance measured in $\Delta_\tau(p)$ defined in equation 4.3.

The first thing to notice for POS is that the performance of the model suffers by 4% on clean data. The performances against intrude and disemvowel improves with increasing attack levels. For attack level *mid* the performance against intrude improves to 20% and for attack level *high* to 23%. The same counts for the disemvowel attack with an improvement to 11% for attack level *mid* and 15% for attack level *high* respectively. This improvement is significantly higher compared to the remaining ones. Considering the models' performances against the full-shuffle, inner-shuffle, truncate and natural-noise attack it is notable that they behave very similar for all attack levels. There is no or just a small improvement, e.g. full-shuffle with attack level *high* to 3%. More conspicuous are the results against the visual and inner-shuffle attack which indicate an increase for attack level *mid* to 5% and 1% and a decrease for attack level *high* to -5% and -2%. The performance against the phonetic attack remains mostly unchanged for all attack levels.

For NLI the performance against the visual attack is the only one that improves on the *low* attack level by 3%. The remaining performances stay the same or even deteriorate. Except for disemvowel, the performances of *mid* and *high* against all attackers are somehow around zero so that no clear statement can be made on the impact of the pretraining process. The performance against disemvowel improves linearly by 8% for each attack level increase starting on 0% improvement on attack level *low*.

The strong impact on the performance against disemvowel does not apply to the TC task. On *low* the same small performance increase as to clean data can be recorded, which becomes relative as the attack level increases. The performance against full-swap behaves in a similar way. In relation to the performance achieved by the POS model against the intrude attack, no significant improvements can be observed for the *low* and *mid* attack level. Whereas an improvement to 9% is accomplished in the *high* attack level.

The model improves its performances against both visual and truncate attack on the *mid* attack level. While the performance against truncate remains on an improvement of 11% at attack level *high*, the performance against visual increases further to 18% total improvement. The performances against the remaining attackers do not change much individually for attack level *mid* and *high*.

5. Discussion

In chapter 4 we have shown the impacts of different attackers to the models' performance on the three individual downstream tasks POS, NLI and TC. In this chapter we analyse and discuss them.

5.1. Attacks

Compared to the majority of the other attackers the phonetic attack is less effective. This score remains the same even if the attack level is increased. We assume the low performance is because of the small changes applied due to the few possibilities to properly perturb a word phonetically. This is because some words do not have a phonetic perturbation. Therefore they cannot be replaced and the count of perturbed samples per sentence remains mostly the same while increasing the attack level. To cope with this new ways to generate phonetic similar words need to be investigated.

The truncate attack performed better than the phonetic attack in all three tasks. It achieved roughly twice the performance reductions. However, we expected results to go beyond that. We suppose this is due to the fact that we chose the number of characters that are truncated too small. This leads to too minor changes that do not fool the model into a wrong decision.

The two sentence-level tasks were also evaluated against the segmentation attack. For TC the performance is almost identically low as the phonetic attack while NLI suffers more, but not as much as expected. The low performance decreases in both tasks which seems to result from *RoBERTa*s' BPE[1] tokenizer. The model is therefore able to split the segmented words back into its parts, i.e. the original words and their subwords, to gain a rough understanding of the original sentence. Thereby the meaning of the original words remain understandable for the model. We ascribe the difference in performance to the occurrence of natural segmentation errors in the training data.

The keyboard-typo attack behaves similar to segmentation with respect to the low performance in the TC task. We conduct this behavior to the same reasons as for segmentation.

As with the segmentation and keyboard-typo the natural-noise attack shows similar low performance decreases for TC. Again we assume the clear contrast depends on natural occurrences of this kind of perturbation in TCs' original train dataset, because it is crawled from real world examples, e.g. social media comments. Therefore it seems like the TC dataset somehow applies unintended adversarial training to the model and makes it perform clearly better in this attack scenario compared to the other tasks. Additionally the fact that the attack performs well for both POS and NLI suggests that the attack is able to perform appropriately. Therefore either the task or the data are responsible for this behavior. In particular the data can be excluded as a reason if each sample the model is trained on will be manually checked beforehand

[1] **B**yte-**P**air **E**ncoding relies on subword units (Y. Liu et al., 2019)

for segmentations, keyboard-typos or other natural-noise. However, this involves considerable effort and therefore needs further investigation.

Both full- and inner-shuffle lead to a significant performance loss. However, full-shuffle performs significantly better than inner-shuffle. This performance difference seems to be related to the first and last letter and therefore to the count of letters which are shuffled. A possible explanation is that the first and last letter have a special influence for the models' understanding of the respective word and therefore on the overall performance of the task. Additional the *high*est attack level does not further improve the performance of inner-shuffle. We assume the attack level increment does not affect the success of the inner-shuffle attack beyond the *mid* attack level, because its maximum has already been achieved.

Disemvowel performs as expected and reduces the performance of the model for each task.

Intrude is one of the two best performing attackers for all three tasks. We suspect this is due to the TC dataset often consisting of long texts with several sentences. After perturbing the data, the words are unknown to the tokenizer and are therefore encoded based on their characters. This means that the number of tokens increases significantly. Thereby *RoBERTa*'s maximum sequence length is exceeded and the end of the input is cut off. This may cause the model to lose important information. This effect also scales with the attack level and would therefore explain the strong drop in performance for the *high*est level.

In addition to intrude, the visual attacker is also one of the best performing attackers. The strong performance decreases indicate that the model has lost its understanding and is therefore assigning the majority label to each input. Additionally the visual attacker replaces letters with ones rarely used in the English language. The BPE tokenizer mostly encodes these letters as two tokens and thereby the same problem occurs as described above.

In Table 5.1 a ranking for the attacks is given. We rank them by the performance degradation added to the model for each individual task. In addition we also report the readability annotated by humans. Therefore we asked 5 non-native English speakers to rank 10 sentences (see Appendix A.9) by their readability. Each of these sentences was perturbed by a different attacker beforehand. It is clear to us that no definite statement can be made from this, but we would like to put the machine performances in some relation to human understanding. Not surprisingly and as above mentioned the visual and the intrude attackers are always the both best performing, followed by full-shuffle ranked 3rd. Except for intrude this applies equally to humans. For visual that was unexpected, because according to Eger et al. (2019) we assumed it to be better readable by humans. We asked the experimentees about their reasons for this particular decision and found out that the main reason was that the visual example contained multiple unknown characters that had to be deciphered. The phonetic attack yields the worst performance for all tasks. This also corresponds to the human annotation.

5.2. Robustness

Adversarial Training: 1-1 Adversarial training on perturbed data, as it is often used in the NLP literature, increases the robustness against every single attacker. This effect equally applies to all tasks. Additionally, it can be said that the robustness increase is tightly bound to the effectiveness of the respective attacker. This means that the robustness increases are generally larger against strong attackers. We assume this is because it is easier to achieve an average performance than SOTA performance. That means, the closer the attackers performance decreases are to SOTA performance the lower the possible robustness increases. Especially for NLI it is striking that the robustness increases are somewhat the inverted shapes of the

Ranking	POS	NLI	TC	Human
1	Intrude	Visual & Intrude	Visual	Visual & Keyboard-Typo
2	Visual	-	Intrude	-
3	Full-Shuffle	Full-Shuffle	Full-Shuffle	Full-Shuffle
4	Keyboard-Typo	Disemvowel	Disemvowel	Natural-Noise
5	Disemvowel	Keyboard-Typo	Inner-Shuffle	Disemvoweling
6	Natural-Noise	Inner-Shuffle	Truncate	Intrude & Inner-Shuffle
7	Inner-Shuffle	Natural-Noise	Keyboard-Typo	-
8	Truncate	Segment	Natural-Noise	Truncate
9	Phonetic	Truncate	Segment	Segment
10	-	Phonetic	Phonetic	Phonetic

Table 5.1.: Ranking on harmfulness of the attackers on POS, NLI, TC and humans on attack level *high*.

performance decreases accomplished by the attackers which supports our assumption. Irregularities only occur for POS with the intrude attack, in which the robustness increase against the *high* attack is smaller than the one for the *mid* attack. For attack level *low* and *mid* this shielding approach compensates almost the entire performance decrease of the attack. This behaviour neither occurs in the other tasks nor for the remaining attackers and is therefore explicitly related to POS and intrude. We assume the shielding loses its effectiveness if the data is perturbed too much in this specific attack scenario.

Adversarial Training: leave-one-out Adversarial training in the leave-one-out fashion, where the model is trained on data perturbed by any attacker except the one, it is evaluated on, adds a reasonably good amount of robustness to all tasks against almost all attackers. As already stated in the 1-1 adversarial training, the robustness of POS against the intrude and visual attacks drops considerably on attack level *high* compared to the other levels. We did not expect such behavior and it does not occur for intrude in the other tasks. The robustness against the visual attack shows similar behavior in the NLI task, whereby it first strongly increases and then drops to a lower total robustness increase. We assume attack level *low* is nearly the maximum which the model is able to handle with this kind of shielding. That's why the robustness increase flattens for both higher attack levels. This is in contrast to TC where the robustness against both attacks increases as expected. Additionally, the robustness against the segment attack decreases for attack level *high* in the NLI task. A similar behavior can be observed in the TC task at which the robustness against the truncate attack suffers of a robustness decrease for attack level *low*. Both before mentioned behaviors were unexpected. Apart from this, we expected better robustness increases for full- and inner-shuffle which are more similar to those of the 1-1 adversarial training. That's because these two attacks are identical except for the inclusion of the first and last letter.

Perturbed Language Model The pretraining on the PLM task yields different robustness changes against the indivdual attackers to the model. Especially for POS the robustness suffers significantly for the majority of attackers. This indicates a negative effect of the shielding approach. For NLI we can make no clear statement about the trend of the robustness, because the changes fluctuate around zero. The robustness in TC overall only increases slightly against all attackers. Especially the robustness against intrude and visual increases significantly. We assume that this is due to their good attack performances. Interestingly the robustness against truncate increases for both POS and TC just as much to neutralize the effectiveness of the attack. It therefore gets somewhat fully robust to this kind of attack.

Perturbed Pretraining Training the model on perturbed data on the MLM taks beforehand leads to poorer results on clean data in the POS task. We suppose the reason therefore is that the model partly overwrites its contextualized word embeddings. This assumption is based on the fact that we pretrain the model on a dataset without completely unchanged samples. Additionally this effect only occurs on the word-based task which relies a lot more on word embeddings. Besides that the robustness only increases for intrude and disemvowel. This is somewhat surprising to us, because the robustness does not equally increase for all perturbations. We expected the same behavior at least against the visual and full-shuffle attackers which also performed above average in the attack scenario. We assume that the samples generated by the aforementioned attackers vary too much and are therefore hard to defend against. For the disemvowel attack this also applies to the NLI task. In the TC task the robustness against most of the attackers increase. This behavior is similar to that observed in the PLM shielding approach.

Comparison In the following we are going to compare the different shielding approaches with each other. One significant property across all approaches is TCs' increasing performance on clean data. This effect maximizes with the perturbed pretraining method. As already mentioned in the interpretation of the low performance of the natural-noise and segment attacks, we assume that the TC dataset already contains perturbed samples and therefore benefits above average. Another indication is that this does not occur in one of the other two tasks. If this is confirmed we can conclude that labeling real world data without removing all types of errors can improve the performances on respective tasks. Besides that the robustness against the phonetic attack has never improved. This result does not indicate bad robustness against this kind of attack. It is rather a result of the unsuccessful attacks (4.1) since the best performance of this model is around 96% accuracy and it is difficult to beat SOTA performance while under attack. In order to get a deeper insight, the experiments need to be repeated after the development of a stronger attack has been finished.

In Figure 5.2 the defenses are ranked by their average robustness increase for each task. Therefore we took the sum over all Δ_τ and normalized it by the number of data points taken into account. Not surprisingly the Adversarial Training 1-1 (AT 1-1) performs best, because its good performance has been proven already in other researches. The biggest pitfall with this approach is that all perturbations have to be known before the training process. This is in particular a very unrealistic scenario through the diversity of perturbations that exists and is therefore only applicable to a limited extend. This problem is addressed by Adversarial Training Leave-One-Out (AT LOO) which achieves the second best robustness increase. Although the perturbation have not yet been seen in training the model is able to achieve better performance compared to the non-shielded case. This is an interesting finding, because we can conclude from it that a model that is trained on different attackers is also able to handle unknown perturbations generated by an additional attacker to a limited extend. For POS and NLI the third and the fourth best performing shielding approaches do not differ significantly. These are Perturbed Pretraining (PP) on rank 3 and PLM on the last place. Neither approach leads to a noteworthy performance increases for the two evaluation tasks. This is in contrast to TC where the last two shielding approaches are ranked in the reverse order.

Ranking	POS		NLI		TC	
1	AT 1-1	+16%	AT 1-1	+20%	AT 1-1	+12%
2	AT LOO	+4%	AT LOO	+10%	AT LOO	+9%
3	PP	+1%	PP	+1%	PLM	+8%
4	PLM	-1%	PLM	0%	PP	+6%

Table 5.2.: Different defense approaches ranked by the average robustness improvement over all attackers.

6. Conclusion

In this work, we introduced ten different low-level adversarial attackers including a newly developed phonetic attacker which is able to replace words by ones sounding similar with the help of sequence-to-sequence systems and phonetic word embeddings. This catalogue only includes a limited number of perturbations, which in no way claims to be complete. We evaluated the SOTA deep-learning model *RoBERTa* against the proposed attacks on different word- and sentence-based tasks and have shown that the model considerably suffers because of them. Furthermore we compared the attackers to each other, worked out their strengths and weaknesses and ranked them by their harmfulness to indicate the three most effective attackers: visual, intrude and full-shuffle. Moreover, it turned out that the phonetic attack was the most ineffective. We attributed this to the fact that this attacker rarely replaces words, because there are often no phonetically similar words. In order to address the large performance loss and make it more robust against adversarial attacks we shielded the model with Adversarial Training, Perturbed Language Model and Perturbed Pretraining. We have shown that the first approach is effective against all kinds of perturbations even evaluated in a leave-one-out fashion whereby the perturbation the model is evaluated on is not used in the adversarial training process. The other two shielding approaches only increased the robustness for the TC evaluation task. The good performance seemed to result from natural occurrences of perturbations in the original train dataset. Evaluation of the other two tasks resulted neither in performance improvements nor in performance deteriorations. We concluded that this is because the orignal pretrained embeddings that made *RoBERTa* perform admirable in different NLP benchmarks were overwritten to a certain extend.

Future Work In this work a catalog of adversarial attacks have been introduced which can be used to evaluate the performance of existing SOTA models under attack. Furthermore, it needs to be extended to cover a wider range of different perturbations and therefore approach a complete list. With focus on the pitfalls in the generation of proper phonetic similar words new methods have to be investigated to strengthen this attack and make it more realistic and successful. In addition to the presented shielding approaches and analogous to visual informed character embeddings (Eger et al., 2019), our phonetic word embeddings can also be used in different models' encoding layer to increase the robustness.

Bibliography

Alzantot, Moustafa et al. (Apr. 2018). 'Generating Natural Language Adversarial Examples'. In: pp. 2890–2896. DOI: 10.18653/v1/d18-1316. URL: http://arxiv.org/abs/1804.07998.

Belinkov, Yonatan and Yonatan Bisk (Nov. 2017). 'Synthetic and Natural Noise Both Break Neural Machine Translation'. In: URL: http://arxiv.org/abs/1711.02173.

Chollet, François et al. (2015). *Keras*. https://keras.io.

Conneau, Alexis et al. (May 2018). 'Supervised Learning of Universal Sentence Representations from Natural Language Inference Data'. In: pp. 670–680. DOI: 10.18653/v1/d17-1070. URL: http://arxiv.org/abs/1705.02364.

Cui, Leyang and Yue Zhang (Aug. 2019). 'Hierarchically-Refined Label Attention Network for Sequence Labeling'. In: pp. 4113–4126. DOI: 10.18653/v1/d19-1422. URL: http://arxiv.org/abs/1908.08676%20http://dx.doi.org/10.18653/v1/D19-1422.

Dai, Falcon and Zheng Cai (2018). 'Glyph-aware Embedding of Chinese Characters'. In: pp. 64–69. DOI: 10.18653/v1/w17-4109. URL: http://github.com/falcondai/chinese-char-lm.

Dehaene, Stanislas and Laurent Cohen (June 2011). *The unique role of the visual word form area in reading*. DOI: 10.1016/j.tics.2011.04.003.

Ebrahimi, Javid, Daniel Lowd and Dejing Dou (2018). 'On Adversarial Examples for Character-Level Neural Machine Translation'. In: URL: https://www.perspectiveapi.com%20http://arxiv.org/abs/1806.09030.

Ebrahimi, Javid, Anyi Rao et al. (Dec. 2018). 'Hotflip: White-box adversarial examples for text classification'. In: *ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*. Vol. 2, pp. 31–36. ISBN: 9781948087346. URL: http://arxiv.org/abs/1712.06751.

Eger, Steffen et al. (Mar. 2019). 'Text Processing Like Humans Do: Visually Attacking and Shielding NLP Systems'. In: URL: http://arxiv.org/abs/1903.11508.

Gil, Yotam et al. (Apr. 2019). 'White-to-Black: Efficient Distillation of Black-Box Adversarial Attacks'. In: pp. 1373–1379. DOI: 10.18653/v1/n19-1139. URL: http://arxiv.org/abs/1904.02405.

Goodfellow, Ian J., Jonathon Shlens and Christian Szegedy (Dec. 2014). 'Explaining and Harnessing Adversarial Examples'. In: URL: http://arxiv.org/abs/1412.6572.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). 'LSTM can solve hard long time lag problems'. In: *Advances in Neural Information Processing Systems*. ISBN: 0262100657.

Hosseini, Hossein et al. (Feb. 2017). 'Deceiving Google's Perspective API Built for Detecting Toxic Comments'. In: URL: http://arxiv.org/abs/1702.08138.

Iyyer, Mohit et al. (Apr. 2018). 'Adversarial Example Generation with Syntactically Controlled Paraphrase Networks'. In: pp. 1875–1885. DOI: 10.18653/v1/n18-1170. URL: http://arxiv.org/abs/1804.06059.

Jia, Robin and Percy Liang (July 2017). 'Adversarial examples for evaluating reading comprehension systems'. In: *EMNLP 2017 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, pp. 2021–2031. ISBN: 9781945626838. DOI: 10.18653/v1/d17-1215. URL: http://arxiv.org/abs/1707.07328.

Jin, Di et al. (July 2019). 'Is BERT Really Robust? A Strong Baseline for Natural Language Attack on Text Classification and Entailment'. In: URL: http://arxiv.org/abs/1907.11932.

Lai, Guokun et al. (Apr. 2017). 'RACE: Large-scale ReAding comprehension dataset from examinations'. In: *EMNLP 2017 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, pp. 785–794. ISBN: 9781945626838. DOI: 10.18653/v1/d17-1082. URL: http://arxiv.org/abs/1704.04683.

Larsen, Anders Boesen Lindbo et al. (Dec. 2016). 'Autoencoding beyond pixels using a learned similarity metric'. In: *33rd International Conference on Machine Learning, ICML 2016*. Vol. 4, pp. 2341–2349. ISBN: 9781510829008. URL: http://arxiv.org/abs/1512.09300.

Liang, Bin et al. (Apr. 2018). 'Deep text classification can be fooled'. In: *IJCAI International Joint Conference on Artificial Intelligence*. Vol. 2018-July, pp. 4208–4215. ISBN: 9780999241127. DOI: 10.24963/ijcai.2018/585. URL: http://arxiv.org/abs/1704.08006%20http://dx.doi.org/10.24963/ijcai.2018/585.

Liu, Frederick et al. (2017). 'Learning character-level compositionality with visual features'. In: *ACL 2017 - 55th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*. Vol. 1. Association for Computational Linguistics (ACL), pp. 2059–2068. ISBN: 9781945626753. DOI: 10.18653/v1/P17-1188.

Liu, Yinhan et al. (July 2019). 'RoBERTa: A Robustly Optimized BERT Pretraining Approach'. In: URL: http://arxiv.org/abs/1907.11692.

Malmasi, Shervin and Marcos Zampieri (Dec. 2017). 'Detecting hate speech in social media'. In: *International Conference Recent Advances in Natural Language Processing, RANLP*. Vol. 2017-Septe, pp. 467–472. ISBN: 9789544520489. DOI: 10.26615/978-954-452-049-6-062. URL: http://arxiv.org/abs/1712.06427.

Merity, Stephen et al. (Sept. 2016). 'Pointer sentinel mixture models'. In: *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*. URL: http://arxiv.org/abs/1609.07843.

Papernot, Nicolas et al. (Apr. 2016). 'Crafting adversarial input sequences for recurrent neural networks'. In: *Proceedings - IEEE Military Communications Conference MILCOM*, pp. 49–54. ISBN: 9781509037810. DOI: 10.1109/MILCOM.2016.7795300. URL: http://arxiv.org/abs/1604.08275.

Pruthi, Danish, Bhuwan Dhingra and Zachary C. Lipton (2019). 'Combating Adversarial Misspellings with Robust Word Recognition'. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Stroudsburg, PA, USA: Association for Computational Linguistics, pp. 5582–5591. DOI: 10.18653/v1/p19-1561. URL: https://www.aclweb.org/anthology/P19-1561.

Rajpurkar, Pranav, Robin Jia and Percy Liang (June 2018). 'Know what you don't know: Unanswerable questions for SQuAD'. In: *ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*. Vol. 2, pp. 784–789. ISBN: 9781948087346. DOI: 10.18653/v1/p18-2124. URL: http://arxiv.org/abs/1806.03822.

Rajpurkar, Pranav, Jian Zhang et al. (June 2016). 'SQuad: 100,000+ questions for machine comprehension of text'. In: *EMNLP 2016 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, pp. 2383–2392. ISBN: 9781945626258. DOI: 10.18653/v1/d16-1264. URL: https://arxiv.org/abs/1606.05250.

Rayner, Keith et al. (Mar. 2006). 'Raeding wrods with jubmled lettres: There is a cost'. In: *Psychological Science* 17.3, pp. 192–193. ISSN: 09567976. DOI: 10.1111/j.1467-9280.2006.01684.x. URL: http://journals.sagepub.com/doi/10.1111/j.1467-9280.2006.01684.x.

Ribeiro, Marco Tulio, Sameer Singh and Carlos Guestrin (2018). 'Semantically equivalent adversarial rules for debugging NLP models'. In: *ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*. Vol. 1, pp. 856–865. ISBN: 9781948087322. DOI: 10.18653/v1/p18-1079.

Rodriguez, Nestor and Sergio Rojas-Galeano (Jan. 2018). 'Shielding Google's language toxicity model against adversarial attacks'. In: URL: http://arxiv.org/abs/1801.01828.

Shimada, Daiki, Ryunosuke Kotani and Hitoshi Iyatomi (2016). 'Document classification through image-based character embedding and wildcard training'. In: *Proceedings - 2016 IEEE International Conference on Big Data, Big Data 2016*, pp. 3922–3927. ISBN: 9781467390040. DOI: 10.1109/BigData.2016.7841067.

Szegedy, Christian et al. (Dec. 2014). 'Intriguing properties of neural networks'. In: *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*. URL: http://arxiv.org/abs/1312.6199.

Wang, Alex et al. (Apr. 2019). 'Glue: A multi-task benchmark and analysis platform for natural language understanding'. In: *7th International Conference on Learning Representations, ICLR 2019*. DOI: 10.18653/v1/w18-5446. URL: http://arxiv.org/abs/1804.07461.

Wang, Xiaosen, Hao Jin and Kun He (Sept. 2019). 'Natural Language Adversarial Attacks and Defenses in Word Level'. In: URL: http://arxiv.org/abs/1909.06723.

Wolf, Thomas et al. (2019). 'HuggingFace's Transformers: State-of-the-art Natural Language Processing'. In: *ArXiv* abs/1910.03771.

Zhang, Wei Emma et al. (Jan. 2019). 'Adversarial Attacks on Deep Learning Models in Natural Language Processing: A Survey'. In: URL: https://arxiv.org/abs/1901.06796%20http://arxiv.org/abs/1901.06796.

Zhang, Zhuosheng et al. (Sept. 2019). 'Semantics-aware BERT for Language Understanding'. In: URL: http://arxiv.org/abs/1909.02209.

Zhao, Zhengli, Dheeru Dua and Sameer Singh (Oct. 2018). 'Generating natural adversarial examples'. In: *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*. URL: http://arxiv.org/abs/1710.11342.

Zhou, Ming et al. (Jan. 2020). *Progress in Neural NLP: Modeling, Learning, and Reasoning*. DOI: 10.1016/j.eng.2019.12.014.

Webpage References

Appendix:English dialect-dependent homophones (2019). URL: https://en.wiktionary.org/wiki/Appendix:English_dialect-dependent_homophones (visited on 11/12/2019).

Appendix:English dialect-independent homophones (2019). URL: https://en.wiktionary.org/wiki/Appendix:English_dialect-independent_homophones (visited on 11/12/2019).

Beal, Vangie (2019). *Huge List of Texting and Online Chat Abbreviations*. URL: https://www.webopedia.com/quick_ref/textmessageabbreviations.asp (visited on 11/12/2019).

Belinkov, Yonatan and Yonatan Bisk (2020). *Github Repository containing scripts and noise data for (Belinkov and Bisk, 2017)*. URL: https://github.com/ybisk/charNMT-noise (visited on 27/02/2020).

ConversionAI-Jigsaw (2020). *Kaggle Toxic Comment Classification Challenge*. URL: https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge (visited on 20/01/2020).

EnglishClub.com (2019). *Homophones List*. URL: https://www.englishclub.com/pronunciation/homophones-list.html (visited on 12/12/2019).

Fitt, S, K Richmond and R Clark (2020). *Combilex*. URL: http://www.cstr.ed.ac.uk/research/projects/combilex/ (visited on 01/01/2020).

Homonym: List of 300+ Homonyms in English with Examples (2019). URL: https://7esl.com/homonyms/ (visited on 12/12/2019).

Homophones List (2019). URL: http://homophonelist.com/homophones-list/ (visited on 13/12/2019).

Jigsaw Toxic Comment Classification Leaderboard (2020). URL: https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/leaderboard (visited on 20/01/2020).

Lundh, Fredrik, Clark Alex and Contributors (2020). *pillow - the friendly PIL fork*. URL: https://python-pillow.org/ (visited on 01/01/2020).

Miller, Ian (2019). *Homophones*. URL: http://www.singularis.ltd.uk/bifroest/misc/homophones-list.html (visited on 12/12/2019).

Multinyms Triple/Quadruple/Quintuple/Sextuple Homonyms (30th Oct. 2015). URL: https://web.archive.org/web/20160825095711/http://people.sc.fsu.edu/~jburkardt/fun/wordplay/multinyms.html (visited on 12/12/2019).

Nordquist, Richard (16th July 2019). *200 Homonyms, Homophones, and Homographs*. URL: https://www.thoughtco.com/homonyms-homophones-and-homographs-a-b-1692660 (visited on 13/12/2019).

Possle, Heiko (2019). *Text Message — SMS — E-Mail — Chat*. URL: https://www.smart-words.org/abbreviations/text.html (visited on 10/12/2019).

Ruder, Sebastian (2020). *NLP-Progress*. URL: https://nlpprogress.com/ (visited on 22/03/2020).

Sainburg, Tim (2020). URL: https://github.com/timsainb/tensorflow2-generative-models (visited on 20/01/2020).

Stanford Natural Language Inference Website (2020). URL: https://nlp.stanford.edu/projects/snli/ (visited on 22/02/2020).

A. Appendix

A.1. Embedding Plots

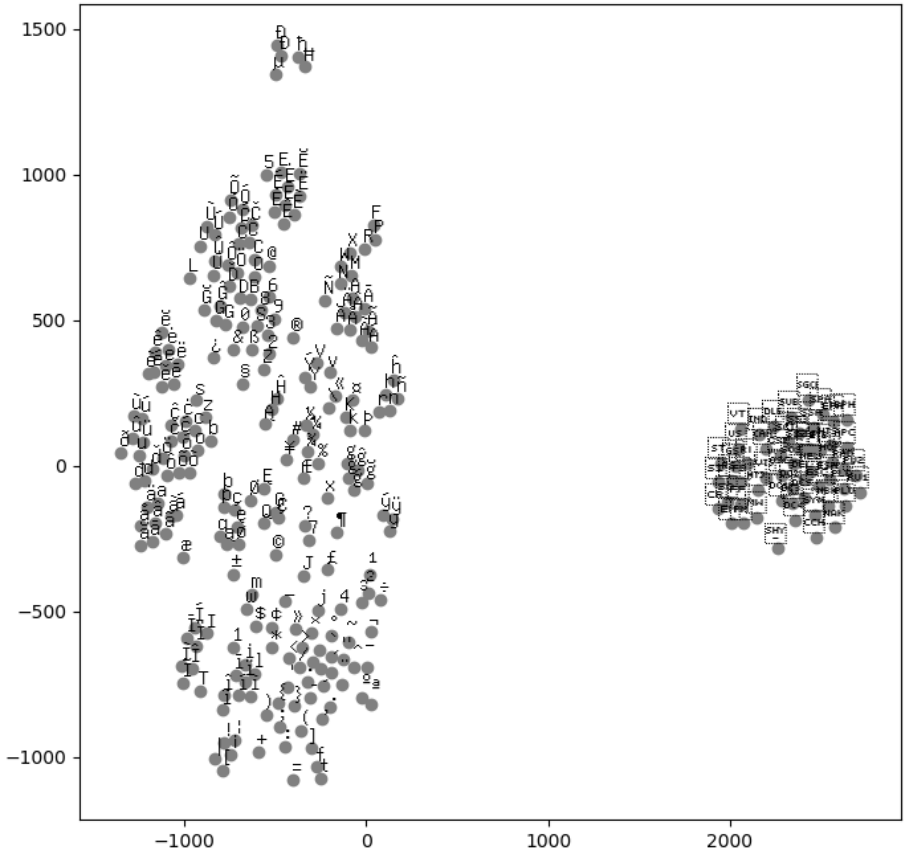


Figure A.1.: t-distributed stochastic neighbor embedding (t-SNE) plot for the embeddings generated by the VAEGAN.

A.2. Result Tables

Acronym	Long
FS	Full-Shuffle
IS	Inner-Shuffle
INT	Intrude
DIS	Disemvowel
TRUN	Truncate
SEG	Segmentation
KEY	Keyboard-Typo
NAT	Natural-Noise
PH	Phonetic
VIS	Visual

Table A.1.: Acronyms used in the later to shorten the tables.

Attack	Mode	Accuracy		AUCROC
		POS	NLI	TC
None	-	96.65	90.41	0.93
Full-Swap	low	82.14	70.35	0.90
	mid	58.14	45.70	0.83
	high	40.47	38.35	0.74
Inner-Swap	low	85.96	67.70	0.90
	mid	70.53	53.35	0.83
	high	67.95	51.55	0.82
Intrude	low	81.42	75.97	0.91
	mid	46.91	52.25	0.85
	high	18.15	34.70	0.66
Disemvowel	low	85.24	72.24	0.91
	mid	61.50	51.62	0.86
	high	44.69	41.00	0.79
Truncate	low	88.57	79.83	0.90
	mid	77,40	72.87	0.84
	high	75,11	72.02	0.83
Segment	low	-	86.08	0.93
	mid	-	77.53	0.92
	high	-	69.14	0.91
Keyboard-Typo	low	85.06	76.93	0.92
	mid	62.41	60.21	0.88
	high	40.99	44.16	0.84
Natural Noise	low	85.34	78.43	0.92
	mid	65.36	65.60	0.91
	high	50.06	56.31	0.90
Phonetic	low	90.62	87.40	0.93
	mid	89.09	84.75	0.92
	high	88.95	82.80	0.91
Visual	low	80.52	53.07	0.86
	mid	48.14	35.26	0.64
	high	22.44	34.37	0.48

Table A.2.: Attacks against unshielded model.

Test \ Train	level	FS	IS	INT	DIS	TRUN	KEY	NAT	PH	VIS
Clean	-	95.17	95.18	95.04	95.44	95.48	95.40	95.40	96.06	95.66
FS	low	**88.49**	81.68	80.01	82.63	81.85	81.45	80.93	82.16	83.10
	mid	**75.73**	58.04	54.04	62.17	59.95	57.69	59.62	56.91	60.38
	high	**62.27**	40.95	36.11	48.06	44.84	40.58	44.40	38.79	42.44
IS	low	87.97	**91.98**	86.06	86.68	85.65	87.09	85.63	86.10	87.70
	mid	74.23	**87.24**	71.42	74.77	72.93	74.06	70.94	70.61	74.61
	high	71.72	**86.24**	69.20	73.20	70.47	71.84	68.54	67.67	72.73
INT	low	79.23	80.38	**92.72**	81.82	82.05	86.88	82.27	79.26	86.72
	mid	39.12	42.02	**88.41**	57.68	54.17	59.29	53.62	44.37	60.84
	high	10.46	11.93	**80.96**	27.95	24.74	29.55	47.71	15.88	22.45
DIS	low	82.29	82.45	84.68	**93.31**	85.91	85.43	85.72	84.27	85.59
	mid	56.75	60.08	62.94	**89.23**	67.55	66.68	67.18	60.33	64.77
	high	40.30	45.11	47.45	**86.13**	53.89	53.37	54.08	42.87	49.31
TRUN	low	88.17	87.50	88.34	88.97	**94.58**	89.45	90.03	88.63	88.80
	mid	77.18	76.02	78.33	79.58	**93.27**	80.58	82.33	76.79	78.98
	high	74.91	73.84	76.28	77.90	**92.99**	78.64	80.84	74.33	76.83
KEY	low	83.70	83.16	85.17	85.36	85.30	**90.11**	85.97	84.11	86.99
	mid	60.52	60.99	63.81	65.23	64.07	**73.45**	67.06	60.62	66.53
	high	36.92	38.50	44.95	44.25	45.64	**62.99**	47.71	37.72	46.93
NAT	low	83.92	82.57	85.21	85.13	84.74	85.83	**88.52**	84.41	86.48
	mid	65.05	62.50	65.83	68.13	66.39	67.97	**80.98**	64.59	67.51
	high	47.83	47.01	52.39	53.52	54.10	54.26	**66.76**	48.79	53.25
PH	low	88.82	88.66	88.89	89.40	89.21	88.86	89.81	**94.89**	89.47
	mid	86.68	86.91	87.12	87.64	87.36	86.93	88.07	**94.39**	87.67
	high	86.74	86.79	87.00	87.51	87.26	86.85	88.00	**94.36**	87.56
VIS	low	82.07	82.13	83.10	82.64	82.61	84.21	82.26	81.20	**87.95**
	mid	53.15	54.79	56.05	57.27	55.55	58.06	53.23	49.32	**66.58**
	high	23.90	26.06	22.19	30.31	29.04	30.43	26.25	23.66	**31.38**

Table A.3.: POS adversarial training: 1-1.

Test \ Train	level	FS	IS	INT	DIS	TRUN	SEG	KEY	NAT	PH	VIS
Clean	-	87.54	-	88.29	88.59	88.91	89.90	89.17	89.12	90.24	-
FS	low	**83.04**	-	64.65	63.28	60.38	62.67	69.46	68.46	66.87	-
	mid	**78.58**	-	47.62	48.15	42.21	46.05	52.63	50.81	48.11	-
	high	**76.96**	-	42.75	44.52	39.16	41.55	47.77	46.21	41.53	-
IS	low	81.31	-	71.32	66.40	59.81	63.26	72.23	72.25	66.40	-
	mid	78.82	-	64.82	58.42	51.17	53.63	63.21	64.24	57.16	-
	high	78.08	-	64.06	58.25	50.86	53.53	62.72	62.89	56.61	-
INT	low	76.22	-	**85.83**	72.49	72.97	72.78	83.73	80.83	74.83	-
	mid	58.99	-	**82.61**	53.87	51.09	48.45	69.53	62.84	51.53	-
	high	43.22	-	**80.76**	39.93	36.18	36.60	48.77	40.91	37.07	-
DIS	low	79.14	-	76.27	**86.56**	67.51	72.52	78.96	77.77	72.65	-
	mid	72.47	-	67.43	**84.70**	57.60	56.88	69.72	64.85	57.03	-
	high	69.45	-	63.90	**84.16**	54.50	48.25	65.29	58.44	48.92	-
TRUN	low	81.63	-	84.31	79.66	**88.15**	80.02	86.35	84.79	80.46	-
	mid	77.87	-	82.45	75.86	**87.52**	76.11	84.08	81.83	76.19	-
	high	77.25	-	82.46	75.10	**87.44**	75.79	83.80	81.76	75.80	-
SEG	low	82.32	-	84.32	83.75	84.00	**89.07**	86.15	85.53	85.69	-
	mid	68.85	-	76.41	75.84	77.33	**87.54**	80.28	79.37	78.14	-
	high	50.94	-	64.98	68.11	71.36	**86.42**	73.39	73.19	71.88	-
KEY	low	74.23	-	76.90	71.38	69.95	73.10	**86.63**	81.04	74.46	-
	mid	57.37	-	62.86	55.62	54.30	58.67	**82.98**	70.74	56.98	-
	high	45.87	-	52.26	46.29	44.75	47.07	**79.82**	61.76	44.54	-
NAT	low	77.87	-	78.32	73.62	73.50	75.51	82.39	**87.67**	76.47	-
	mid	67.98	-	67.98	62.27	60.10	62.97	74.25	**85.45**	62.85	-
	high	59.73	-	60.81	55.16	53.18	55.41	69.52	**84.06**	54.89	-
PH	low	85.68	-	85.98	84.23	85.36	86.53	87.50	87.80	**89.93**	-
	mid	84.25	-	84.21	80.98	82.67	84.17	85.98	86.50	**89.40**	-
	high	83.07	-	82.71	80.28	81.68	82.68	84.74	85.42	**89.19**	-
VIS	low	59.79	-	72.33	55.74	56.09	56.91	70.65	66.32	55.34	-
	mid	41.82	-	50.95	37.87	37.01	36.38	45.21	41.84	36.31	-
	high	37.42	-	39.08	33.81	34.26	34.51	36.04	35.37	34.10	-

Table A.4.: NLI adversarial training: 1-1.

Test \ Train	level	FS	IS	INT	DIS	TRUN	SEG	KEY	NAT	PH	VIS
Clean	-	0.96	0.96	0.96	0.97	0.97	0.98	0.97	0.96	0.97	0.96
FS	low	**0.94**	0.94	0.94	0.94	0.94	0.95	0.95	0.93	0.95	0.93
	mid	**0.92**	0.90	0.87	0.88	0.87	0.87	0.89	0.87	0.87	0.88
	high	**0.90**	0.86	0.80	0.81	0.79	0.79	0.83	0.80	0.77	0.83
IS	low	0.94	**0.95**	0.94	0.94	0.94	0.95	0.94	0.93	0.94	0.93
	mid	0.92	**0.94**	0.89	0.90	0.89	0.89	0.91	0.88	0.89	0.89
	high	0.91	**0.94**	0.88	0.90	0.88	0.88	0.90	0.87	0.88	0.89
INT	low	0.95	0.95	**0.96**	0.96	0.96	0.96	0.96	0.95	0.96	0.95
	mid	0.92	0.92	**0.95**	0.92	0.92	0.91	0.94	0.90	0.91	0.91
	high	0.81	0.80	**0.91**	0.80	0.76	0.75	0.81	0.76	0.72	0.83
DIS	low	0.94	0.95	0.95	**0.96**	0.94	0.96	0.95	0.94	0.95	0.94
	mid	0.91	0.91	0.91	**0.96**	0.90	0.90	0.92	0.90	0.89	0.90
	high	0.88	0.88	0.88	**0.95**	0.86	0.83	0.89	0.86	0.83	0.88
TRUN	low	0.95	0.95	0.96	0.96	**0.97**	0.96	0.97	0.94	0.96	0.95
	mid	0.94	0.94	0.95	0.94	**0.97**	0.95	0.96	0.93	0.95	0.94
	high	0.94	0.94	0.95	0.94	**0.97**	0.95	0.96	0.93	0.95	0.94
SEG	low	0.96	0.96	0.96	0.96	0.97	**0.97**	0.97	0.96	0.97	0.95
	mid	0.95	0.95	0.95	0.96	0.96	**0.97**	0.96	0.95	0.96	0.94
	high	0.94	0.94	0.94	0.95	0.95	**0.97**	0.95	0.93	0.95	0.93
KEY	low	0.95	0.95	0.95	0.95	0.96	0.96	**0.96**	0.95	0.96	0.95
	mid	0.92	0.92	0.93	0.93	0.93	0.94	**0.95**	0.92	0.93	0.92
	high	0.88	0.88	0.91	0.89	0.90	0.89	**0.95**	0.89	0.87	0.90
NAT	low	0.96	0.96	0.96	0.96	0.97	0.97	0.97	**0.96**	0.97	0.95
	mid	0.95	0.95	0.96	0.96	0.96	0.97	0.97	**0.96**	0.96	0.95
	high	0.95	0.95	0.95	0.95	0.96	0.96	0.96	**0.96**	0.96	0.94
PH	low	0.96	0.96	0.96	0.96	0.97	0.97	0.97	0.95	**0.97**	0.95
	mid	0.95	0.95	0.95	0.96	0.96	0.97	0.96	0.95	**0.97**	0.95
	high	0.95	0.95	0.95	0.95	0.96	0.96	0.96	0.94	**0.97**	0.94
VIS	low	0.92	0.93	0.94	0.93	0.93	0.93	0.94	0.91	0.92	**0.93**
	mid	0.82	0.81	0.86	0.80	0.78	0.77	0.83	0.76	0.71	**0.90**
	high	0.70	0.69	0.75	0.65	0.62	0.62	0.66	0.64	0.55	**0.85**

Table A.5.: TC adversarial training: 1-1.

Test \ Train	level	FS	IS	INT	DIS	TRUN	SEG	KEY	NAT	PH	VIS	POS	NLI	TC
FS	none	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	95.57	89.56	0.97
	low											84.49	73.05	0.95
	mid											63.48	57.54	0.90
	high											45.72	51.73	0.86
IS	none	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	95.66	88.94	0.96
	low											88.29	75.51	0.94
	mid											75.90	69.07	0.91
	high											73.68	68.54	0.90
INT	none	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	95.65	88.90	0.96
	low											87.54	84.27	0.95
	mid											57.58	74.92	0.93
	high											19.44	61.07	0.84
DIS	none	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	95.69	89.42	0.96
	low											86.00	80.00	0.94
	mid											64.39	70.98	0.91
	high											48.65	66.60	0.89
TRUN	none	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	95.49	89.17	0.96
	low											89.98	84.55	0.84
	mid											81.23	81.97	0.83
	high											79.38	81.62	0.82
SEG	none	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	-	89.02	0.96
	low											-	85.38	0.96
	mid											-	76.92	0.95
	high											-	62.83	0.95
KEY	none	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	95.61	88.80	0.96
	low											87.71	80.47	0.95
	mid											68.64	70.69	0.94
	high											46.51	61.83	0.92
NAT	none	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	95.72	88.67	0.96
	low											88.17	81.27	0.96
	mid											72.30	73.05	0.96
	high											56.78	67.40	0.96
PH	none	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	95.30	88.95	0.96
	low											89.74	87.54	0.96
	mid											87.82	86.27	0.95
	high											87.72	85.34	0.95
VIS	none	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	95.72	89.02	0.96
	low											85.18	70.77	0.93
	mid											58.94	48.80	0.85
	high											24.99	40.22	0.75

Table A.6.: Adversarial training: leave-one-out.

Attack	Mode	Accuracy		AUCROC
		POS	NLI	TC
None	-	92.47	90.62	0.97
Full-Swap	low	80.13	65.38	0.93
	mid	60.20	45.82	0.83
	high	42.85	41.78	0.73
Inner-Swap	low	84.48	64.86	0.93
	mid	72.73	54.25	0.86
	high	70.55	54.16	0.85
Intrude	low	84.18	73.75	0.95
	mid	65.99	50.57	0.89
	high	39.66	36.67	0.74
Disemvowel	low	84.88	71.40	0.94
	mid	71.79	58.21	0.86
	high	62.06	54.86	0.78
Truncate	low	86.44	78.53	0.95
	mid	78.12	74.40	0.93
	high	76.40	74.18	0.93
Segment	low	-	86.07	0.97
	mid	-	80.00	0.96
	high	-	75.02	0.95
Keyboard-Typo	low	82.68	74.82	0.95
	mid	63.33	56.76	0.91
	high	42.39	44.56	0.86
Natural Noise	low	83.21	76.42	0.96
	mid	64.56	62.51	0.96
	high	50.56	54.49	0.95
Phonetic	low	87.35	87.25	0.96
	mid	85.87	84.56	0.96
	high	85.74	83.03	0.95
Visual	low	80.19	55.43	0.91
	mid	52.41	36.77	0.75
	high	19.45	34.16	0.65

Table A.7.: Attacks against model pretrained on perturbed data and afterwards finetuned on clean data.

Attacker	Mode	Accuracy		AUCROC
		POS	NLI	TC
None	-	96.18	89.92	0.97
Full-Swap	low	82.80	68.00	0.94
	mid	49.43	47.65	0.85
	high	28.69	39.75	0.76
Inner-Swap	low	87.24	65.31	0.93
	mid	62.75	52.12	0.87
	high	58.91	51.07	0.86
Intrude	low	83.14	72.50	0.96
	mid	40.23	47.13	0.91
	high	18.06	37.84	0.81
Disemvowel	low	89.68	71.01	0.94
	mid	75.60	53.00	0.88
	high	64.24	45.08	0.82
Truncate	low	91.23	78.66	0.96
	mid	83.06	72.05	0.94
	high	80.82	71.34	0.94
Segment	low	-	85.24	0.97
	mid	-	78.86	0.96
	high	-	72.90	0.95
Keyboard-Typo	low	86.95	75.40	0.96
	mid	57.96	58.79	0.92
	high	33.05	45.13	0.88
Natural Noise	low	87.11	76.46	0.97
	mid	65.25	63.23	0.96
	high	47.33	55.14	0.95
Phonetic	low	91.32	86.82	0.97
	mid	89.60	83.75	0.96
	high	89.52	82.56	0.95
Visual	low	75.29	48.38	0.92
	mid	30.72	37.67	0.79
	high	08.55	35.06	0.68

Table A.8.: Attacks against model pretrained on pertubed language model task and afterwards finetuned on clean data.

A.3. Human Annotation

Perturber	Sentence	H1	H2	H3	H4	H5	∅
FS	iThs hcchur hoirc ssing to hte ssemas as ehty sgni suooyj gsons rofm teh boko at a chuhcr.	10	9	10	3	6	7,6
IS	Tihs cruchh coihr sngis to the mseass as tehy snig jooyus snogs form the book at a crchuh.	3	7	9	2	5	5,2
INT	T,hi,s c{hu{r{c{h c[h[o[i[r si<ngs t]o t:h:e m.a.s.s.e.s a@s th?e?y si'n'g j{oy{o{u{s s_o_n_g_s fr:o:m t.h.e book a-t a ch'urc'h.	5	2	3	9	7	5,2
DIS	Ths chrch chr sngs to th msss as thy sng jys sngs frm th bk at a chrch.	9	6	5	7	4	6,2
TRU	Thi churc choi sing to the masse as the sin joyou song fro the boo at a churc.	4	4	4	5	3	4
KEY	Thid chudch chlir singz go hhe masces az they wing moyous somgs vrom yhe book qt w chuech.	7	10	8	6	9	8
SEG	Thischurchchoirsingstothemassesastheysingjoyoussongsfromthebookata church.	2	3	2	4	2	2,6
PH	This church choir sings to the masses as they sing Jayoo's songs from the book at a church.	1	1	1	1	1	1
VIS	Ʈhɯš ʰṻʳch cḧòir sŏ̈ńṡ tȍ ṯḫe ɯáṡėṡ àṣ ẗhɘẏ sĩnʒ ʲöẏǭūs ʂongʂ ƒrøᵑ tȟê 𝒝őôk aƚ ấ ḫ̥űɹĆħ.	8	8	6	10	8	8
NAT	Ths curch choir sings tu th? masses asa thy sig joyous sons [[fron el buck ay as thurch.	6	5	7	8	10	7,2

Table A.9.: Human Readability Experiment where Hn indicate the readability rank given by the respective human annotator. 1 indicate the best readability and 10 the worst.

Erklärung zur Abschlussarbeit gemäß §22 Abs. 7 und §23 Abs. 7 APB der TU Darmstadt

Hiermit versichere ich, Yannik Benz, die vorliegende Bachelorarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Fall eines Plagiats (§38 Abs. 2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei der abgegebenen Thesis stimmen die schriftliche und die zur Archivierung eingereichte elektronische Fassung gemäß §23 Abs. 7 APB überein.

Bei einer Thesis des Fachbereichs Architektur entspricht die eingereichte elektronische Fassung dem vorgestellten Modell und den vorgelegten Plänen.

Darmstadt, den 14. April 2020

Yannik Benz