IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 25, 2021, accepted January 9, 2022, date of publication February 11, 2022, date of current version February 17, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3148413

A Differentiable Language Model Adversarial Attack on Text Classifiers

IVAN FURSOV[1], ALEXEY ZAYTSEV[1], PAVEL BURNYSHEV[1,2], EKATERINA DMITRIEVA[3], NIKITA KLYUCHNIKOV[1], ANDREY KRAVCHENKO[4], EKATERINA ARTEMOVA[2,3], EVGENIA KOMLEVA[1], AND EVGENY BURNAEV[1,5]

[1]Skolkovo Institute of Science and Technology, 121205 Moscow, Russia
[2]Huawei Noah's Ark Laboratory, 620073 Moscow, Russia
[3]HSE University, 101000 Moscow, Russia
[4]University of Oxford, Oxford OX1 2JD, U.K.
[5]Artificial Intelligence Research Institute (AIRI), 103001 Moscow, Russia

Corresponding author: Alexey Zaytsev (a.zaytsev@skoltech.ru)

This work was supported by the Analytical Center through the RF Government (Subsidy Agreement 000000D730321P5Q0002) under Grant 70-2021-00145.

ABSTRACT Transformer models play a crucial role in state of the art solutions to problems arising in the field of natural language processing (NLP). They have billions of parameters and are typically considered as black boxes. Robustness of huge Transformer-based models for NLP is an important question due to their wide adoption. One way to understand and improve robustness of these models is an exploration of an adversarial attack scenario: check if a small perturbation of an input invisible to a human eye can fool a model. Due to the discrete nature of textual data, gradient-based adversarial methods, widely used in computer vision, are not applicable per se. The standard strategy to overcome this issue is to develop token-level transformations, which do not take the whole sentence into account. The semantic meaning and grammatical correctness of the sentence are often lost in such approaches In this paper, we propose a new black-box sentence-level attack. Our method fine-tunes a pre-trained language model to generate adversarial examples. A proposed differentiable loss function depends on a substitute classifier score and an approximate edit distance computed via a deep learning model. We show that the proposed attack outperforms competitors on a diverse set of NLP problems for both computed metrics and human evaluation. Moreover, due to the usage of the fine-tuned language model, the generated adversarial examples are hard to detect, thus current models are not robust. Hence, it is difficult to defend from the proposed attack, which is not the case for others. Our attack demonstrates the highest decrease of classification accuracy on all datasets(on AG news: 0.95 without attack, 0.89 under SamplingFool attack, 0.82 under DILMA attack).

INDEX TERMS Deep learning, natural language processing, adversarial attack, generative model.

I. INTRODUCTION

For the last decade deep learning has been an extremely active and growing area of research. It provides solutions for various problems in computer vision and natural language processing. Whilst useful, deep learning models are hard to explain, as they have billions of parameters and hence are black boxes with no robustness guarantees. Adversarial attacks [1] in all application areas including computer vision [2], [3], natural language processing [4]–[6], and graphs [7] seek to reveal non-robustness of deep learning models.

The associate editor coordinating the review of this manuscript and approving it for publication was Alicia Fornés.

An adversarial attack on a text classification model perturbs the input sentence in such a way that the deep learning model is fooled, while the perturbations adhere to certain constraints, utilising semantic similarity, morphology or grammar patterns. The deep learning model then misclassifies the perturbed sentence, whilst for a human it is evident that the sentence's class remains the same [8].

Unlike for images and their pixels, for textual data it is not possible to estimate the derivatives of class probabilities with respect to input tokens due to the discrete nature of the language and its vocabulary. Although a sentence or a word representation can lie in a continuous space, the token itself can not be altered slightly to get to the neighbouring point. This turns partial derivatives useless.

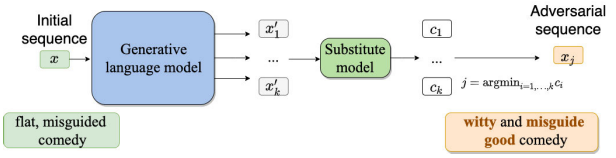

FIGURE 1. The workflow of an adversarial attack. With a generative model we create a sample of possibly adversarial sequences. Then, we select the best one via a substitute model. This example is adversarial with high probability: a genuine model gives a wrong answer for it, whilst it is also similar to the initial sequence.

Many adversarial attacks that accept the initial space of tokens as input attempt to modify these sequences using operations like addition, replacement, or switching of tokens [9]–[11]. Searching for the best modification can be stated as a discrete optimisation problem, which often appears to be computationally hard and is solved by random or greedy search heuristics in practice [11]. Otherwise gradient optimisation techniques can be leveraged in the embedding space [12]–[14]. This approach has a bottleneck: all information contained in a sentence has to be squeezed into a single sentence embedding.

To alleviate these problems we propose a new differentiable adversarial attack model, which benefits from leveraging pre-trained language models, DILMA (DIfferentiable Language Model Attack). A general outline of our approach is presented in Figure 1. The proposed model for an adversarial attack has two regimes. Both regimes work in a black-box model without knowledge of gradients of an attack, while the second regime uses a substitute (surrogate [15]) model output to make attack more efficient.

The first regime is a random sampling from a pre-trained language model that produces adversarial examples by chance, whilst constraining discrepancy between an original and a modified sequence. The second regime is a targeted attack that modifies the language model by optimising the loss with two terms related to misclassification by the target model and a discrepancy between an initial sequence and its adversarial counterpart. After the generation of a set of sequences we select the most adversarial sequence according to the substitute model scores.

Thus, we expect that by construction the generated adversarial examples will fool a deep learning model, but will remain semantically close to the initial sequence. For the first regime, we don't need to fine-tune a language model, whilst in some cases we will require to sample many sequences to find an adversarial one. For the second regime, the model fine-tuning requires some computational time, but after the parameters are fixed our approach produces adversarial examples with high success rate.

For the training phase in the second regime, we use a separately trained differentiable version of the Levenshtein distance [16] and the Gumbel-Softmax heuristic to pass the derivatives through our sequence generation layers. As our loss is differentiable, we can adopt any gradient-based adversarial attack. The number of hyperparameters in our method is small, as we want our attack to be easily adopted to new datasets and problems. The training and inference procedure is summarised in Figure 2. Examples of sentences generated via our attack DILMA are presented in Table 1.

To summarise, the main contributions of this work are the following.

- We propose a new black-box adversarial attack DILMA on classifier models for texts. The attack is based on a masked language model (MLM) and a differentiable loss function to optimise during an attack. The model utilises a substitute classifier.
- Our DILMA relies on fine-tuning parameters of an MLM by optimising the weighted sum of two differentiable terms: one is based on a surrogate distance between the source text and its adversarial version, and another is a substitute classifier model score (Sec. III). Hence, we adopt a generative model framework for the task of the generation of adversarial examples.
- We apply DILMA to various NLP sentence classification tasks and receive superior results in comparison to other methods (Sec. IV).
- We show that a particular advantage of DILMA, when compared to other approaches, lies in the resistance to common defense strategies. The vast majority of existing approaches fail to fool models after they defend themselves (Sec. V).
- We provide a framework to extensively evaluate adversarial attacks for sentence classification and conduct a thorough evaluation of DILMA and related attacks. Adversarial training and adversarial detection must be an essential part of the adversarial attack evaluation on textual data in support to human evaluation (Sec. V-D).

II. RELATED WORK

There exist adversarial attacks for different types of data: the most mainstream being image data [17], [18], graph data [19], and sequences [20]. The latter work is one of the first publications on generation of adversarial attacks for natural language processing, such as texts. It identifies two main challenges for the task: a discrete space of possible objects and a complex definition of a semantically coherent sequence.

There is a well-established categorisation of textual attacks according to the perturbation level. **Character-level** attacks include replacing some of the characters with visually similar ones [21]. HotFlip uses gradient-based decisions to swap, remove, or insert characters [11]. [22] creates sub-word perturbations to craft non-standard English adversaries and mitigates the cultural biases of machine translation models. **Token-level** attacks replace a token (mostly at the word level) in a given sentence, based on a salience score and following a specific synonym replacement strategy. [23] chooses the best synonym replacement from WordNet to generate a sentence, close to the classifier's decision boundary. [24] and [25] utilise BERT's masked model to generate the substitution to the target word in multiple ways. The Metropolis-Hastings

TABLE 1. Attack samples for the PWWS attack (top performing according to our evaluation) and our attack DILMA. DILMA can provide more diverse adversarial sequences with meanings similar to that of the initial sentence.

Dataset	Original	PWWS	DILMA (ours)
rotten tomatoes	flat, misguided comedy	unconditional, misadvise comedy	witty and misguide good comedy
	more likely to have you scratching your head than hiding under your seat	more potential to have you scraping your forefront than hiding under your seat	more likely to have you scratching your head than brood beneath your seat
	a rehash of every gangster movie from the past decade	a retrograde of every gangster movie from the past decade	a rehash of every good film of the past decade
sst2	this is a story of two misfits who dont stand a chance alone but together they are magnificent	this is a floor of two misfits who dont stall a chance alone but unitedly they are magnificent	this is a story of two misfits who do not stand a chance alone but together they are united
	the draw for big bad love is a solid performance by arliss howard	the draw for gravid bad love is a square performance by arliss howard	the song for big bad love is a 1964 performance by arliss howard
	a gripping movie played with performances that are all understated and touching	a gripping movie played with performances that are all downplay and advert	a horror movie with lyrics that are all understated and demanding
dstc	i wanna play the movie	i wanna frolic the movie	i wanna play the games
	could you help me call a cab to the airport?	could you help me visit a cabriolet to the airport?	could you help me find a taxi to the airport?
	i would like to book 1 room at a nearby hotel.	i would like to book single room at a nearby hotel.	i would like to book 1 room at a approved hotel.

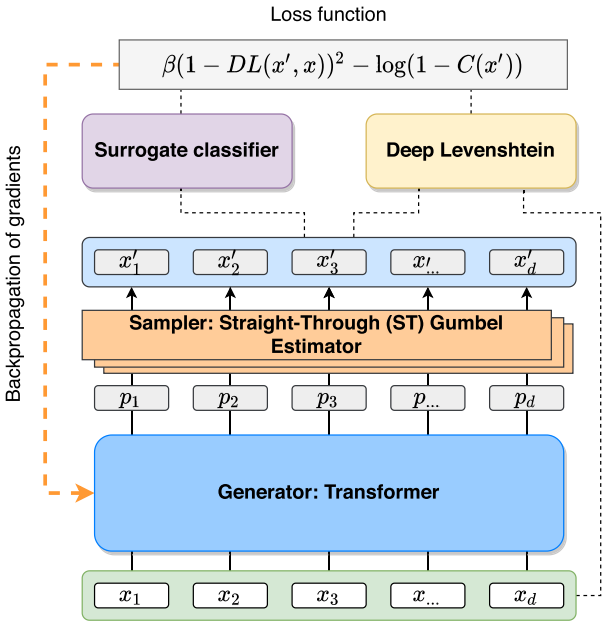

FIGURE 2. The training phase of the DILMA architecture consists of the following steps. Step 1: obtain logits P from a pre-trained Language Model (LM) generator for input x. Step 2: sample x′ from the multinomial distribution P using the Gumbel-Softmax estimator. To improve generation quality we can sample many times. Step 3: obtain the substitute probability $C(x')$ and approximate the edit distance $DL(x', x)$. Step 4: calculate the loss, do a backward pass. Step 5: update parameters of the LM using the computed gradients.

algorithm improves sampling from a constrained distribution, allowing to substitute a word in a sentence which belongs to a desired class [26]. **Sentence-level** attacks aim to generate a new adversarial instance from scratch with the help of paraphrasing models [27], back translation [28] or competitive dialogue agents [29].

Another way to categorise textual adversarial attacks accounts for the amount of information received from the victim target model. **White box** attacks have full access to the victim model and its gradients. Generating adversarial examples guided by the training loss is intractable due to the discrete nature of textual data. Instead greedy methods [30] or the Gumbel trick [31] are used to make computations tractable. **Black box** attacks have access only to the model outputs. Victim's scores may help to estimate word salience [32]. Black box attacks can emulate white box attacks. For example, DistFlip [33] is a faster version of HotFlip with a similar accuracy.

Blind attacks do not have access to the target model at all and can be seen as a form of text augmentation. Such attacks include stop word removal, dictionary-based word replacement, and rule-based insertion of negation [34].

Methods to defend against attacks exploit adversarial stability training [35] and robust word and character representations [36]. Back-off strategies to recognise intentionally replaced and corrupted words can detect perturbed inputs before actually passing them through a classifier [37].

Common tasks requiring defence from adversarial attacks by adversarial training, include classification, sentence pair problems, such as natural language inference (NLI) [32], and machine translation [38], [39].

From a technical point of view, two open source frameworks for textual adversarial attacks, adversarial training and text augmentation, namely, `OpenAttack` [40] and `TextAttack` [6], facilitate the research in the area and help to conduct fast and correct comparison.

From the current state of the art, we see a lack of effective ways to generate adversarial categorical sequences and defend from such attacks. Existing approaches use previous generations of LMs based on recurrent architectures, stay at a token level, or use VAEs, despite known limitations of these models for modelling NLP data [41], [42].

III. METHODS

A. GENERAL DESCRIPTION OF THE APPROACH

We generate adversarial examples using two consecutive components: a masked language model (MLM) with parameters $\boldsymbol{\theta}$ that provides for an input sequence $\mathbf{x}$, conditional distribution $p_{\boldsymbol{\theta}}(\mathbf{x}'|\mathbf{x})$, and a sampler from this distribution such that $\mathbf{x}' \sim p_{\boldsymbol{\theta}}(\mathbf{x}'|\mathbf{x})$. Thus, we can generate sequences $\mathbf{x}'$ by a consecutive application of the MLM and the sampler.

We can use a pre-trained MLM to generate probably adversarial sequences. But to make sampling more efficient, we optimise MLM parameters $\boldsymbol{\theta}$ with respect to a proposed differentiable loss function that forces the MLM to generate semantically similar but adversarial examples. The loss function consists of two terms: the first term corresponds to a substitute classifier $C(\mathbf{x}')$ that outputs a probability of belonging to a target class, and the second term corresponds to a Deep Levenshtein distance $DL(\mathbf{x}, \mathbf{x}')$ that approximates the edit distance between sequences. Thus, by minimising the loss function, we expect to get low scores for a substitute classifier, fooling the model, but keep semantics similar by minimising an approximation of Levenshtein distance between an initial sequence and an adversarial one.

The general scheme of our approach is given in Fig. 2. We start with the description of the LM and the sampler in Subsection III-B. We continue with the description of the loss function in Subsection III-C. The formal description of our algorithm is given in Subsection III-D. In later subsections III-E and III-F, we provide more details on the use of the target and substitute classifiers and the Deep Levenshtein model correspondingly.

B. MASKED LANGUAGE MODEL

The MLM is a model that takes a sequence of tokens (e.g. words) as input $\mathbf{x} = \{x_1, \ldots, x_t\}$ and outputs logits for tokens $\mathbf{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_t\} \in \mathbb{R}^{d \cdot t}$ for each index $1, \ldots, t$, where d is the size of the dictionary of tokens. In this work we use a transformer architecture as the LM [42]. We pre-train a transformer encoder [42] in a BERT [43] manner from scratch using the available data.

The sampler is defined as follows: the MLM learns a probability distribution over sequences, so we can sample a new sequence $\mathbf{x}' = \{x'_1, \ldots, x'_t\}$ based on the vectors of token logits $\mathbf{P}$. In this work we use Straight-Through Gumbel Estimator $ST(\mathbf{P}) : \mathbf{P} \to \mathbf{x}'$ for sampling [44]. To get the actual probabilities q_{ij} from logits p_{ij}, we use a softmax with temperature τ [45], where $\tau > 1$, which produces a softer probability distribution over classes. As $\tau \to \infty$, the original distribution approaches the uniform distribution. If $\tau \to 0$, then sampling from it reduces to arg max sampling.

Our first method **SamplingFool** samples sequences from the categorical distribution with q_{ij} probabilities. The method is similar to the random search algorithm and serves as a baseline.

C. LOSS FUNCTION

The Straight-Through Gumbel sampling allows a propagation of the derivatives through the softmax layer. Hence, we optimise parameters $\boldsymbol{\theta}$ of our MLM to improve the quality of generated adversarial examples.

Let $C_y^t(\mathbf{x})$ be the probability of the target classifier to predict the considered class y. A loss function takes two terms into account: the first term estimates the probability score drop $(1-C_y^t(\mathbf{x}'))$ of the target classifier, the second term represents the edit distance $DL(\mathbf{x}, \mathbf{x}')$ between the initial sequence $\mathbf{x}$ and the generated sequence $\mathbf{x}'$. We should maximise the probability drop and minimise the edit distance, so it is as close to 1 as possible.

In the black-box scenario we do not have access to the true classifier score, so we use a substitute classifier score $C_y(\mathbf{x}') \approx C_y^t(\mathbf{x}')$. As a differentiable alternative to edit distance, we use the Deep Levenshtein model proposed in [16] — a deep learning model that approximates the edit distance: $DL(\mathbf{x}, \mathbf{x}') \approx D(\mathbf{x}, \mathbf{x}')$.

This way, the loss function becomes:

$$L(\mathbf{x}', \mathbf{x}, y) = \beta(1 - DL(\mathbf{x}', \mathbf{x}))^2 - \log(1 - C_y(\mathbf{x}')), \quad (1)$$

where $C_y(\mathbf{x}')$ is the probability of the true class y for sequence $\mathbf{x}'$ and β is a weighting coefficient. Thus, we penalise cases when a modification in more than one token is needed in order to get $\mathbf{x}'$ from $\mathbf{x}$. Since we focus on non-target attacks, the $C_y(\mathbf{x}')$ component is included in the loss. The smaller the probability of an attacked class, the smaller the loss.

We estimate derivatives of $L(\mathbf{x}', \mathbf{x}, y)$ in a way similar to [46]. Using these derivatives, a backward pass updates the weights $\boldsymbol{\theta}$ of the MLM. We find that updating the whole set of parameters $\boldsymbol{\theta}$ is not the best strategy, and a better alternative is to update only the last linear layer and the last layer of the generator.

We consider two DILMA options: initial **DILMA** that minimises only the classifier score (the second term) during the attack and **DILMA with DL** that takes into account the approximate Deep Levenshtein edit distance.

D. DILMA ALGORITHM

Now we are ready to group the introduced components into a formal algorithm with the architecture depicted in Fig. 2.

Given a sequence $\mathbf{x}$, the **DILMA** attack performs the following steps at iterations $i = 1, 2, \ldots, k$ starting from the parameters of a pre-trained MLM $\boldsymbol{\theta}_0 = \boldsymbol{\theta}$ and running for k iterations.

Step 1. Pass the sequence $\mathbf{x}$ through the pre-trained MLM. Obtain logits $\mathbf{P} = LM_{\boldsymbol{\theta}_{i-1}}(\mathbf{x})$.

Step 2. Sample an adversarial sequence $\mathbf{x}'$ from the logits using the Gumbel-Softmax Estimator.

Step 3. Calculate the probability $C_y(\mathbf{x}')$ and the Deep Levenshtein distance $DL(\mathbf{x}', \mathbf{x})$. Calculate the loss value $L(\mathbf{x}', \mathbf{x}, y)$ (1).

Step 4. Do a backward pass to update the LM's weights $\boldsymbol{\theta}_{i-1}$ using gradient descent and get the new weights $\boldsymbol{\theta}_i$.

Step 5. Obtain an adversarial sequence $\mathbf{x}'_i$ by sampling based on the softmax with a selected temperature.

The algorithm chooses which tokens should be replaced. The classification component changes the sequence in a direction where the probability score $C_y(\mathbf{x}')$ is low, and the Deep Levenshtein distance keeps the generated sequence close to the original one.

We obtain a set of m sampled adversarial sequences on each iteration of the algorithm. The last sequence in this set is not always the best one. Therefore among generated sequences that are adversarial w.r.t. the substitute classifier $C_y(\mathbf{x}')$ we choose $\mathbf{x}'_{\text{opt}}$ with the lowest Word Error Rate (WER), estimated with respect to the initial sequence $\mathbf{x}$. If all examples don't change the model prediction w.r.t. $C_y(\mathbf{x}')$, then we select $\mathbf{x}'_{\text{opt}}$ with the smallest target class score, estimated by the substitute classifier.

We choose the hyperparameter set achieving the best NAD score with the Optuna framework [47].

E. CLASSIFICATION MODEL

In all experiments we use two classifiers: a target classifier $C^t(\mathbf{x})$ that we attack and a substitute classifier $C(\mathbf{x})$ that provides differentiable substitute classifier scores.

The target classifier is RoBerta [48]. The substitute classifier is the LSTM model. The hidden size is 150, the dropout rate is 0.3.

The substitute classifier has access only to 50% of the data, whilst the target classifier uses the entire dataset. We split the dataset into two parts keeping the class balance similar for each part.

F. THE DEEP LEVENSHTEIN MODEL

The differentiable version of the edit distance allows gradient-based updates of parameters. Following the Deep Levenshtein approach [16], we train a deep learning model $DL(\mathbf{x}, \mathbf{x}')$ to estimate the Word Error Rate (WER) between two sequences $\mathbf{x}$ and $\mathbf{x}'$. We treat WER as the word-level Levenshtein distance.

Following [49], the Deep Levenshtein model receives two sequences $(\mathbf{x}, \mathbf{y})$. It encodes them into dense representations $\mathbf{z_x} = E(\mathbf{x})$ and $\mathbf{z_y} = E(\mathbf{y})$ of fixed length l using the shared encoder. Then it concatenates the representations and the absolute difference between them in a vector $(\mathbf{z_x}, \mathbf{z_y}, |\mathbf{z_x} - \mathbf{z_y}|)$ of length $3l$. At the end the model uses a fully-connected layer to predict WER. To estimate the parameters of the encoder and the fully connected layer we use the L_2 loss between the true and the predicted WER values. We form a training sample of size of about two million data points by sampling pairs of sequences and their modifications from the training data.

IV. EXPERIMENTS

The datasets and the source code are published online[1].

[1]The code is available at https://anonymous.4open.science/r/4cf31d59-9fe9-4854-ba53-7be1f9f6be7d/

A. COMPETING APPROACHES

We compared our approach to other popular approaches described below. By selecting the methods among the ones available in the literature, we target the diversity of proposed approaches and focus on the top performers for each direction. We also aim to cover attacks with different trade-offs between the desired word error rate (WER) and the classifier score drop. Whilst this trade-off for some attacks is a result of the selection of hyperparameteres, the exact configuration is not possible, so we focus on such hyperparameters that maximise the NAD score.

HotFlip [11] selects the best token to change, given an approximation of partial derivatives for all tokens and all elements of the dictionary. To change multiple tokens HotFlip selects the sequence of changes via a beam search.

Textbugger [50] works at a symbol level and tries to replace symbols in words to generate new adversarial sequences using derivative values for possible replacements.

DeepWordBug [51] is a greedy replace-one-word heuristic scoring based on an estimate of importance of a word to a RNN model with each replacement being a character-swap attack. DeepWordBug is a black box attack.

PWWS [52] is a greedy synonym-swap method, which takes into account the saliencies of the words and the effectiveness of their replacement. Replacements of the words are made with the help of the WordNet dictionary.

B. DATASETS

We conducted experiments on four open NLP datasets for different tasks such as text classification, intent prediction, and sentiment analysis. The characteristics of these datasets are presented in Table 2.

The AG News corpus (AG) [53] consists of news articles on the web from the AG corpus. There are four classes: World, Sports, Business, and Sci/Tech. Both training and test sets are perfectly balanced. The **Dialogue State Tracking Challenge** dataset (DSTC) is a special processed dataset related to dialogue system tasks. The standard DSTC8 dataset [54] was adopted to the intent prediction task by extracting most intent-interpreted sentences from dialogues. **The Stanford Sentiment Treebank** (SST-2) [55] contains phrases with fine-grained sentiment labels in the parse trees of 11, 855 sentences from film reviews. **The Rotten Tomatoes** dataset (RT) [56] is a film-review dataset of sentences with positive or negative sentiment labels.

C. NAD METRIC

In order to create an adversarial attack, changes must be applied to the initial sequence. A change can be done either by inserting, deleting, or replacing a token in some position in the original sequence. In the *WER* calculation, the cost of any change to the sequence made by insertion, deletion, or replacement is 1. Therefore, we consider the adversarial sequence to be perfect if $WER = 1$, and the target classifier output has changed. For the classification task, Normalised

TABLE 2. Statistics of the datasets.

	Classes	Avg. length	Max length	Train size	Test size	Targeted RoBERTa accuracy	Substitute LSTM accuracy
AG	4	6.61	19	115000	5000	0.87	0.86
DSTC	6	8.82	33	5452	500	0.85	0.85
SST-2	2	8.62	48	62349	5000	0.82	0.80
RT	2	18.44	51	8530	1066	0.76	0.70

TABLE 3. The NAD metric before/after adversarial training on 5,000 examples. We want to maximise the NAD metric for both cases. The best values are in **bold**, the second best values are underscored. DILMA is resistant to adversarial training as well as sampling fool.

	AG	DSTC	SST-2	RT
DeepWordBug	0.017 / **0.012**	0.156 / 0.034	0.12 / 0.077	0.071 / 0.054
HotFlip	0.015 / 0.009	0.09 / 0.084	0.023 / 0.001	0.034 / 0.011
PWWS	0.018 / 0.011	0.142 / 0.025	0.156 / 0.068	**0.102** / **0.072**
TextBugger	0.015 / 0.009	0.098 / 0.023	0.066 / 0.041	0.066 / 0.059
SamplingFool (ours)	0.015 / 0.004	0.218 / 0.113	0.137 / 0.126	0.04 / 0.022
DILMA (ours)	**0.023** / 0.003	0.237 / 0.111	0.19 / 0.154	0.045 / 0.016
DILMA with DL (ours)	0.022 / 0.004	**0.241** / **0.131**	**0.208** / **0.172**	0.052 / 0.02

TABLE 4. The attack success rate and WER The best values are in **bold**, the second best values are underscored. DILMA shows the highest attack success rate among all datasets.

	Attack Success rate				WER			
	AG	**DSTC**	**SST-2**	**RT**	**AG**	**DSTC**	**SST-2**	**RT**
DeepWordBug	0.07	0.30	0.23	0.21	7.68	1.40	1.83	2.59
HotFlip	0.06	0.20	0.08	0.13	**0.37**	**0.37**	**0.07**	**0.13**
PWWS	0.07	0.26	0.22	0.21	6.46	1.16	1.33	1.77
TextBugger	0.07	0.22	0.11	0.17	8.07	1.31	1.33	2.16
SamplingFool (ours)	0.11	0.36	0.28	0.18	14.29	2.59	4.01	7.30
DILMA (ours)	**0.18**	**0.43**	0.43	0.23	17.24	2.86	4.08	7.64
DILMA with DL (ours)	0.14	0.42	**0.47**	**0.24**	12.48	2.49	3.82	7.12

Accuracy Drop (NAD) is calculated in the following way:

$$NAD(A) = \frac{1}{N}\sum_{i=1}^{N}\frac{\mathbf{1}\{C^t(\mathbf{x}_i) \neq C^t(\mathbf{x}'_i))\}}{WER(\mathbf{x}_i, \mathbf{x}'_i)},$$

where $\mathbf{x}' = A(\mathbf{x})$ is the output of an adversarial generation algorithm for the input sequence $\mathbf{x}$, $C^t(\mathbf{x})$ is the label assigned by the target classification model, and $WER(\mathbf{x}', \mathbf{x})$ is the Word Error Rate. The highest value of NAD is achieved when $WER(\mathbf{x}'_i, \mathbf{x}_i) = 1$ and $C(\mathbf{x}_i) \neq C(\mathbf{x}'_i)$ for all i. Here we assume that adversaries produce distinct sequences and $WER(\mathbf{x}_i, \mathbf{x}'_i) \geq 1$.

V. EVALUATION AND COMPARISON

A. ADVERSARIAL ATTACK EVALUATION

Results for the considered methods are presented in Table 3. We demonstrate not only the quality of attacks on the initial target classifier, but also after its re-training with additional adversarial samples added to the training set. After re-training the initial target classifier, competitor attacks cannot provide reasonable results, whilst our methods perform significantly better before and after retraining for most of the datasets. We provide additional attack quality metrics in supplementary materials.

B. ADDITIONAL METRICS FOR EVALUATION

We provide two additional metrics to judge the quality of our and competitors' attacks: **accuracy** of target models changing after attacks and **probability difference** – a difference between true model scores before and after an attack. Higher values for **probability differences** mean that an attack is more successful in affecting the target model decision.

Discriminator training [57] is another defence strategy we followed. We trained an additional discriminator on 10, 000 samples of the original sequences and adversarial examples. The discriminator detects whether an example is normal or adversarial. The discriminator has an LSTM architecture with fine-tuning of GLoVE embeddings and was trained for 50 epochs using the negative log-likelihood loss with early stopping based on validation scores.

We provide all these scores in Table 5. As we can see our methods are top performer with respect to the obtained accuracy scores and probability differences. Due to smaller and more careful changes, HotFlip is harder to detect by a discriminator. However, ROC AUC scores are close to a ROC AUC score of a random classifier for two of four datasets for our attacks.

C. LINGUISTIC EVALUATION

To measure the linguistic features of the generated adversarial sentences we calculated several metrics, which can be divided into two groups: showing how far the generated samples are from the original ones and demonstrating the linguistic quality of adversarial examples.

TABLE 5. Additional performance scores before (first row) and after different attacks. Our methods are in *italic*. The best values are in **bold**, the second best values are underscored.

	Accuracy score				PD metric				ROC AUC for adv discriminator			
	AG	**DSTC**	**SST-2**	**RT**	**AG**	**DSTC**	**SST-2**	**RT**	**AG**	**DSTC**	**SST-2**	**RT**
Before attack	*0.95*	*0.89*	*0.94*	*0.89*								
DeepWordBug	0.93	0.70	0.77	0.79	0.02	0.21	0.18	0.10	0.90	0.92	0.88	0.81
HotFlip	0.94	0.80	0.92	0.87	0.01	0.09	0.02	0.03	**0.43**	0.59	**0.44**	**0.46**
PWWS	0.93	0.74	0.78	0.79	0.02	0.18	0.17	0.11	0.79	0.86	0.64	0.60
TextBugger	0.93	0.78	0.89	0.83	0.02	0.12	0.06	0.06	0.85	0.84	0.68	0.63
SamplingFool (our)	0.89	0.64	0.72	0.82	0.06	0.27	0.23	0.07	0.75	0.44	0.65	0.60
DILMA (our)	**0.82**	**0.57**	0.57	0.77	**0.13**	**0.34**	0.37	0.12	0.81	0.43	0.65	0.55
DILMA with DL (our)	0.86	0.58	**0.53**	**0.76**	0.09	0.33	**0.41**	**0.13**	0.73	**0.43**	0.65	0.52

TABLE 6. Obtained diversity metrics Dist-2 and Ent-2 after an attack. ↑ / ↓ shows if a metric was increased or decreased. Results for the initial sequences are in the first row. All attacks increase the diversity compared to the initial sentences, whilst maintaining semantics of generated adversarial examples.

	AG		**DSTC**		**SST-2**		**RT**	
	Dist	Ent	Dist	Ent	Dist	Ent	Dist	Ent
Before the attack:	0.69	9.81	0.34	7.48	0.72	8.64	0.71	9.30
DeepWordBug	0.75 ↑	9.95 ↑	0.46 ↑	7.90 ↑	0.76 ↑	8.72 ↑	0.75↑	9.37 ↑
HotFlip	0.69	9.82 ↑	0.38 ↑	7.61 ↑	0.72	8.64	0.71	9.30
PWWS	0.73 ↑	9.91 ↑	0.40 ↑	7.75 ↑	0.74 ↑	8.68 ↑	0.73 ↑	9.34 ↑
TextBugger	0.74 ↑	9.99 ↑	0.42	7.80 ↑	0.75 ↑	8.71 ↑	0.73 ↑	9.36 ↑
SamplingFool (our)	0.71 ↑	9.98 ↑	0.43 ↑	7.76 ↑	0.76	8.84 ↑	0.74 ↑	9.38 ↑
DILMA (our)	0.63 ↓	9.53 ↓	0.41 ↑	7.73 ↑	0.73 ↑	8.77↑	0.72 ↑	9.34 ↑
DILMA with DL (our)	0.69	9.89 ↑	0.41 ↑	7.72 ↑	0.74 ↑	8.78↑	0.73 ↑	9.36 ↑

These results show that the generated sentences are not very different from the original ones. The same conclusion also follows from the morphological and syntactic analyses.

We evaluate a **similarity of part-of-speech (POS) annotations** between original and adversarial sentences using the Jaccard index (the intersection of two sets over the union of two sets) between sets containing the POS tags of words from the original A and adversarial B sentences. Duplicated POS-tags in the sets are allowed. In the same way, the **similarity of syntax annotations** is estimated as the Jaccard index of dependency relation tags. We use the Stanza toolkit [58] for POS-tagging and dependency parsing. The detailed report about morphological and syntactic similarities can be found in the Appendix.

The **diversity** across the generated sentences was evaluated by two measures. The first measure, *Dist-k*, is the total number of distinct k-grams divided by the total number of produced tokens in all the generated sentences [59]. The second measure, *Ent-k* [60] considers that infrequent k-grams contribute more to diversity than frequent ones. Table 6 presents the results for $k = 2$. We choose this value as being more indicative to estimate the diversity. A table for other values of k can be found in the Appendix. As we can see, presented methods preserve the lexical diversity.

D. HUMAN EVALUATION

We conducted a human evaluation to understand how comprehensive DILMA adversarial perturbations are and to compare DILMA to other approaches. We used a sample from the SST-2 dataset of size 200, perturbed with five different attacks.

We recruited crowd workers from a crowdsourcing platform to estimate the accuracy of the methods we compare. Given a sentence and the list of classes, the workers were asked to define a class label. We used original sentences to control the workers' performance and estimated the results on adversarial sentences. Each sentence was shown to 5 workers. We used the majority vote as the final label. For all different attacks we achieve similar performances at about 0.70 ± 0.02, with HotFlip scoring 0.72 and DILMA scoring 0.7.

VI. CONCLUSION

Constructing adversarial attacks for natural language processing is a challenging problem due to the discrete nature of input data and non-differentiability of the loss function. Our idea is to combine sampling from a masked language model (MLM) with tuning of its parameters to produce truly adversarial examples. To tune parameters of the MLM we use a loss function based on two differentiable surrogates – for a distance between sequences and for an attacked classifier. This results in the proposed DILMA with DL approach. If we only sample from the MLM, we obtain a simple baseline SamplingFool.

In order to estimate the efficiency of adversarial attacks on categorical sequences we have proposed a metric combining WER and the accuracy of the target classifier. For

TABLE 7. Word error rate (WER) of attacks. Our methods are in *italic*. Smaller values are better. The best values are in **bold**, the second best values are underscored.

	AG	DSTC	SST-2	RT
DeepWordBug	7.69	1.41	1.83	2.60
HotFlip	**0.36**	**0.37**	**0.07**	**0.14**
PWWS	6.46	1.15	1.33	1.77
TextBugger	8.07	1.31	1.33	2.16
SamplingFool (our)	14.30	2.57	4.02	7.31
DILMA (our)	17.24	2.86	4.09	7.65
DILMA with DL (our)	12.48	2.49	3.82	7.13

TABLE 8. ROC AUC scores (↓) for adversarial discriminator as a countermeasure against adversarial attacks: a binary classification "adversary vs non-adversary". Our methods are in *italic*. The best values are in **bold**, the second best values are underscored. Only successful attacks are considered.

	AG	DSTC	SST-2	RT
DeepWordBug	0.930	0.897	0.904	0.885
HotFlip	**0.458**	0.913	**0.444**	0.566
PWWS	0.768	0.761	0.669	0.623
TextBugger	0.816	0.815	0.701	0.672
SamplingFool (our)	0.695	**0.707**	0.716	**0.429**
DILMA (our)	0.863	0.759	0.739	0.568
DILMA with DL (our)	0.699	0.728	0.739	0.496

the considered set of diverse NLP datasets, our approaches demonstrate a good quality outperforming other approaches. The second term in the loss function related to the Deep Levenshtein model is important, as it improves performance scores. Moreover, in contrast to competing methods, our approaches win over common strategies used to defend from adversarial attacks. Human and linguistic evaluation also show the adequacy of the proposed attacks.

VII. SUPPLEMENTARY MATERIALS

A. INTRODUCTION

Supplementary materials consist of the following subsection:

- In subsection VII-B we present additional metrics to assess the overall quality of attacks.
- In subsection VII-C we show whether our attack can be detected in a simple way.
- In subsection VII-E we present additional linguistic metrics to assess the diversity and similarity to the initial sequence for the sequences generated by adversarial attacks.
- In subsection VII-F we make notes about complexity of our method and competing approaches.

B. ADDITIONAL METRICS ON OVERALL ATTACK QUALITY

Word Error Rate (WER) derives from the Levenshtein distance, working at the word level instead of the phoneme level. It is the ratio of the number of changes required to get one token sequence from another divided by the total number of tokens. We use words as tokens in this case.

$$WER = \frac{S + D + I}{N},$$

where S is the number of substitutions, D is the number of deletions, I is the number of insertions, and N is the number of words in the reference. The mean results are presented in Table 7. We see that whilst our attacks change more words than others, they can preserve the original meanings of sentences according to our other experiments.

C. ADDITIONAL EXPERIMENT ON ADVERSARIAL DETECTION

We conducted an additional experiment on the detection of adversarial examples. We train an adversarial discriminator: a classifier that tries to distinguish adversarial and normal sequences for different attacks. Our discriminator is a bidirectional LSTM with GloVe embeddings. We run the training phase for samples of about 1, 000 examples with the number being slightly different for different attacks. The stopping criterion was an increase in the loss for a test sample.

In Table 8, we report ROC AUC values for this classifier. As we want our attack to be undetectable, low ROC AUC values certify that our attack is hard to detect via a model of decent quality. We see, that across different datasets the performance of different attacks differ, while among alternative to ours approach, we observe either weak overall performance (HotFlip) or high detection rate by a discriminator (DeepWordBug, PWWS, TextBugger).

D. ADDITIONAL LINGUISTIC METRICS ON ATTACK QUALITY

Semantic, morphological, and syntactic similarities show how far is the generated sentence from the original one from the linguistic point of view. Detailed results are presented in Table 9.

E. ADDITIONAL LINGUISTIC METRICS ON ATTACK QUALITY

Semantic, morphological, and syntactic similarities show how far is the generated sentence from the original one from the linguistic point of view. Detailed results are presented in Table 9.

The **diversity** of the generated sentences was evaluated by *Dist-k* and *Ent-k* metrics. *Dist-k* is the total number of distinct k-grams divided by the total number of produced tokens in all of the generated sentences. *Ent-k* is the analogue of entropy and considers that infrequent k-grams contribute more to diversity than frequent ones. We consider these metrics for different values of k to get broader evaluation of the diversity of generated sequences. The results are in Table 10 for $Dist - k$ and in Table 11 for $Ent - k$.

F. ADDITIONAL EXPERIMENT DETAILS

The DILMA attack performs 30 iterations on each sample. The output is chosen based on the probability drop of the target model. Competing approaches do not strictly restrict the number of attack iterations. Still, they use a list of constraints

TABLE 9. Semantic (Sem), morphological (Mor), and syntactic (Syn) similarities between original and generated sentences. Our methods are emphasised in *italic*. The values related to closest texts are in **bold**, the second closest values are underscored.

	AG			DSTC			SST-2			RT		
	Sem	Mor	Syn	Sem	Morh	Syn	Sem	Mor	Syn	Sem	Mor	Syn
DeepWordBug	0.61	0.74	0.77	0.74	0.95	0.94	0.51	0.75	0.83	0.68	0.87	0.88
HotFlip	**0.97**	**0.98**	**0.98**	**0.9**	**0.98**	**0.98**	**0.98**	**0.99**	**0.99**	**0.98**	**0.99**	**0.99**
PWWS	0.72	0.83	0.89	0.82	0.97	0.96	0.72	0.89	0.91	0.83	0.94	0.93
TextBugger	0.69	0.83	0.88	0.81	0.96	0.95	0.76	0.92	0.93	0.8	0.94	0.93
SamplingFool (our)	0.65	0.73	0.8	0.73	0.90	0.90	0.52	0.66	0.69	0.67	0.87	0.86
DILMA (our)	0.49	0.61	0.66	0.70	0.87	0.86	0.47	0.63	0.66	0.64	0.83	0.82
DILMA with DL (our)	0.71	0.73	0.78	0.72	0.88	0.88	0.48	0.64	0.67	0.68	0.86	0.85

TABLE 10. Dist-k results after an attack. Results before an attack are in the first row. Our methods are emphasised in *italic*.

	AG			DSTC			SST-2			RT		
k	1	2	4	1	2	4	1	2	4	1	2	4
Before attack	0.19	0.69	0.88	0.09	0.34	0.62	0.32	0.72	0.68	0.24	0.71	0.86
DeepWordBug	0.32	0.75	0.89	0.16	0.46	0.69	0.41	0.76	0.69	0.31	0.75	0.86
HotFlip	0.2	0.69	0.88	0.11	0.38	0.65	0.32	0.72	0.68	0.24	0.71	0.86
PWWS	0.23	0.73	0.89	0.11	0.4	0.67	0.34	0.74	0.69	0.25	0.73	0.86
TextBugger	0.26	0.74	0.89	0.13	0.42	0.68	0.37	0.75	0.69	0.27	0.73	0.86
SamplingFool (our)	0.21	0.71	0.9	0.13	0.43	0.67	0.33	0.76	0.71	0.27	0.74	0.86
DILMA (our)	0.18	0.63	0.86	0.11	0.41	0.67	0.28	0.73	0.71	0.25	0.72	0.86
DILMA with DL (our)	0.2	0.69	0.89	0.11	0.41	0.67	0.29	0.74	0.71	0.26	0.73	0.86

TABLE 11. Ent-k results after an attack. Results before an attack are in the first row. Our methods are emphasised in *italic*.

	AG			DSTC			SST-2			RT		
k	1	2	4	1	2	4	1	2	4	1	2	4
Before attack	7.18	9.81	10.45	5.2	7.48	8.8	6.52	8.64	8.74	6.51	9.3	9.83
DeepWordBug	7.56	9.95	10.48	5.56	7.9	9.01	6.72	8.72	8.77	6.69	9.37	9.83
HotFlip	7.19	9.82	10.46	5.3	7.61	8.87	6.51	8.64	8.74	6.5	9.3	9.83
PWWS	7.35	9.91	10.47	5.4	7.75	8.96	6.58	8.68	8.76	6.56	9.34	9.83
TextBugger	7.49	9.99	10.53	5.48	7.8	8.97	6.67	8.71	8.79	6.64	9.36	9.85
SamplingFool (our)	7.28	9.98	10.56	5.46	7.76	8.97	6.48	8.84	8.93	6.74	9.38	9.85
DILMA (our)	6.66	9.53	10.43	5.33	7.73	8.97	6.29	8.77	8.92	6.56	9.34	9.85
DILMA with DL (our)	7.14	9.89	10.53	5.34	7.72	8.96	6.31	8.78	8.91	6.64	9.36	9.84

as a stopping mechanism: not allowing the attack to modify one word twice, not allowing the attack to alter a word from stoplist, stopping if the Levenshtein distance between source and perturbed sequence crosses a threshold.

VIII. CONCLUSION

The provided metric values support our claims from the main article about a decent quality of our attacks DILMA, DILMA with DL and SamplingFool compared to the attacks on NLP models presented currently in the literature.

REFERENCES

[1] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 9, pp. 2805–2824, Sep. 2019.

[2] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *IEEE Access*, vol. 6, pp. 14410–14430, 2018.

[3] I. Oseledets and V. Khrulkov, "Art of singular vectors and universal adversarial perturbations," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8562–8570.

[4] W. E. Zhang, Q. Z. Sheng, A. Alhazmi, and C. Li, "Adversarial attacks on deep learning models in natural language processing: A survey," 2019, *arXiv:1901.06796*.

[5] W. Wang, R. Wang, L. Wang, Z. Wang, and A. Ye, "Towards a robust deep neural network in texts: A survey," 2019, *arXiv:1902.07285*.

[6] J. Morris, E. Lifland, J. Y. Yoo, J. Grigsby, D. Jin, and Y. Qi, "TextAttack: A framework for adversarial attacks, data augmentation, and adversarial training in NLP," in *Proc. Conf. Empirical Methods Natural Lang. Process., Syst. Demonstrations*, 2020, pp. 119–126.

[7] L. Sun, Y. Dou, C. Yang, J. Wang, P. S. Yu, L. He, and B. Li, "Adversarial attack and defense on graph data: A survey," 2018, *arXiv:1812.10528*.

[8] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," 2016, *arXiv:1611.01236*.

[9] S. Samanta and S. Mehta, "Towards crafting text adversarial samples," 2017, *arXiv:1707.02812*.

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

[11] J. Ebrahimi, A. Rao, D. Lowd, and D. Dou, "HotFlip: White-box adversarial examples for text classification," in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics*, vol. 2, 2018, pp. 31–36.

[12] M. Sato, J. Suzuki, H. Shindo, and Y. Matsumoto, "Interpretable adversarial perturbation in input embedding space for text," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 4323–4330.

[13] Y. Ren, J. Lin, S. Tang, J. Zhou, S. Yang, Y. Qi, and X. Ren, "Generating natural language adversarial examples on a large scale with generative models," 2020, *arXiv:2003.10388*.

[14] I. Fursov, A. Zaytsev, N. Kluchnikov, A. Kravchenko, and E. Burnaev, "Gradient-based adversarial attacks on categorical sequence models via traversing an embedded world," 2020, *arXiv:2003.04173*.

[15] Z. Yan, Y. Guo, and C. Zhang, "Subspace attack: Exploiting promising subspaces for query-efficient black-box attacks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 3825–3834.

[16] S. Moon, L. Neves, and V. Carvalho, "Multimodal named entity recognition for short social media posts," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol., (Long Papers)*, vol. 1, 2018, pp. 852–860.

[17] C. Szegedy, W. Zaremba, I. Sutskever, J. B. Estrach, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *Proc. ICLR*, 2014, pp. 1–10.

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

[19] D. Zügner, A. Akbarnejad, and S. Günnemann, "Adversarial attacks on neural networks for graph data," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 2847–2856.

[20] N. Papernot, P. McDaniel, A. Swami, and R. Harang, "Crafting adversarial input sequences for recurrent neural networks," in *Proc. MILCOM IEEE Mil. Commun. Conf.*, Nov. 2016, pp. 49–54.

[21] S. Eger, G. G. Şahin, A. Rücklé, J.-U. Lee, C. Schulz, M. Mesgar, K. Swarnkar, E. Simpson, and I. Gurevych, "Text processing like humans do: Visually attacking and shielding," in *Proc. Conf. North*, 2019, pp. 1634–1647.

[22] S. Tan, S. Joty, M.-Y. Kan, and R. Socher, "It's Morphin' time! Combating linguistic discrimination with inflectional perturbations," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, 2020, pp. 2920–2935.

[23] Z. Meng and R. Wattenhofer, "A geometry-inspired attack for generating natural language adversarial examples," in *Proc. 28th Int. Conf. Comput. Linguistics*, 2020, pp. 6679–6689.

[24] L. Li, R. Ma, Q. Guo, X. Xue, and X. Qiu, "BERT-ATTACK: Adversarial attack against BERT using BERT," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2020, pp. 6193–6202.

[25] S. Garg and G. Ramakrishnan, "BAE: BERT-based adversarial examples for text classification," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2020, pp. 6174–6181.

[26] H. Zhang, H. Zhou, N. Miao, and L. Li, "Generating fluent adversarial examples for natural languages," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 5564–5569.

[27] W. C. Gan and H. T. Ng, "Improving the robustness of question answering systems to question paraphrasing," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 6065–6075.

[28] Y. Zhang, J. Baldridge, and L. He, "Paws: Paraphrase adversaries from word scrambling," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Human Lang. Technol. (Long and Short Papers)*, vol. 1, 2019, pp. 1298–1308.

[29] M. Cheng, W. Wei, and C.-J. Hsieh, "Evaluating and enhancing the robustness of dialogue systems: A case study on a negotiation agent," in *Proc. Conf. North*, 2019, pp. 3325–3335.

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

[31] P. Yang, J. Chen, C.-J. Hsieh, J.-L. Wang, and M. I. Jordan, "Greedy attack and gumbel attack: Generating adversarial examples for discrete data," *J. Mach. Learn. Res.*, vol. 21, no. 43, pp. 1–36, 2020.

[32] D. Jin, Z. Jin, J. T. Zhou, and P. Szolovits, "Is bert really robust? A strong baseline for natural language attack on text classification and entailment," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, no. 5, 2020, pp. 8018–8025.

[33] Y. Gil, Y. Chai, O. Gorodissky, and J. Berant, "White-to-black: Efficient distillation of black-box adversarial attacks," in *Proc. Conf. North*, 2019, pp. 1373–1379.

[34] T. Niu and M. Bansal, "Adversarial over-sensitivity and over-stability strategies for dialogue models," in *Proc. 22nd Conf. Comput. Natural Lang. Learn.*, 2018, pp. 486–496.

[35] H. Liu, Y. Zhang, Y. Wang, Z. Lin, and Y. Chen, "Joint character-level word embedding and adversarial stability training to defend adversarial text," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, 2020, pp. 8384–8391.

[36] E. Jones, R. Jia, A. Raghunathan, and P. Liang, "Robust encodings: A framework for combating adversarial typos," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, 2020, pp. 2752–2765.

[37] D. Pruthi, B. Dhingra, and Z. C. Lipton, "Combating adversarial misspellings with robust word recognition," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 5582–5591.

[38] M. Cheng, J. Yi, P.-Y. Chen, H. Zhang, and C.-J. Hsieh, "Seq2sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, 2020, pp. 3601–3608.

[39] W. Zou, S. Huang, J. Xie, X. Dai, and J. Chen, "A reinforced generation of adversarial examples for neural machine translation," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, 2020, pp. 3486–3497.

[40] G. Zeng, F. Qi, Q. Zhou, T. Zhang, Z. Ma, B. Hou, Y. Zang, Z. Liu, and M. Sun, "OpenAttack: An open-source textual adversarial attack toolkit," 2020, *arXiv:2009.09191*.

[41] Z. Yang, Z. Hu, R. Salakhutdinov, and T. Berg-Kirkpatrick, "Improved variational autoencoders for text modeling using dilated convolutions," in *ICML*, pp. 3881–3890, 2017.

[42] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. NeurIPS*, 2017, pp. 5998–6008.

[43] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2018.

[44] E. Jang, S. Gu, and B. Poole, "Categorical reparametrization with gumble-softmax," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017.

[45] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," Tech. Rep., 2015.

[46] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," Tech. Rep., 2016.

[47] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 2623–2631.

[48] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," Tech. Rep., 2019.

[49] X. Dai, X. Yan, K. Zhou, Y. Wang, H. Yang, and J. Cheng, "Convolutional embedding for edit distance," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2020, pp. 599–608.

[50] J. Li, S. Ji, T. Du, B. Li, and T. Wang, "TextBugger: Generating adversarial text against real-world applications," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2019, pp. 1–15.

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

[52] S. Ren, Y. Deng, K. He, and W. Che, "Generating natural language adversarial examples through probability weighted word saliency," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 1085–1097.

[53] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," Tech. Rep., 2015.

[54] A. Rastogi, X. Zang, S. Sunkara, R. Gupta, and P. Khaitan, "Schema-guided dialogue state tracking task at DSTC8," 2020, *arXiv:2002.01359*.

[55] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2013, pp. 1631–1642.

[56] B. Pang and L. Lee, "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales," in *Proc. 43rd Annu. Meeting Assoc. Comput. Linguistics (ACL)*, 2005, pp. 115–124.

[57] H. Xu, Y. Ma, H. Liu, D. Deb, H. Liu, J. Tang, and A. K. Jain, "Adversarial attacks and defenses in images, graphs and text: A review," 2019, *arXiv:1909.08072*.

[58] P. Qi, Y. Zhang, Y. Zhang, J. Bolton, and C. D. Manning, "Stanza: A Python natural language processing toolkit for many human languages," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics, Syst. Demonstrations*, 2020, pp. 101–108.

[59] J. Li, M. Galley, C. Brockett, J. Gao, and B. Dolan, "A diversity-promoting objective function for neural conversation models," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, 2016, pp. 110–119.

[60] Y. Zhang, M. Galley, J. Gao, Z. Gan, X. Li, C. Brockett, and B. Dolan, "Generating informative and diverse conversational responses via adversarial information maximization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 1815–1825.

IVAN FURSOV was born in Chelyabinsk, Russia. He received the master's degree from the Skolkovo Institute of Science and Technology, in 2020. He is currently a Deep Learning Research Engineer at Tinkoff and continues to work on new approaches in adversarial attacks on NLP models. In his master's thesis, he proposed a new adversarial attack on sequence classifiers.

ALEXEY ZAYTSEV was born in Kharkiv, Ukraine. He graduated from the MIPT, in 2012. He received the Ph.D. degree in mathematics from the IITP RAS, in 2017. He is currently an Assistant Professor at the Skoltech. His research interests include development of new methods for sequential data, Bayesian optimization, and embeddings for weakly structured data. In his master's thesis, he proposed a modification of Bayesian approach for linear regression that allows an automated feature selection.

PAVEL BURNYSHEV was born in Perm, Russia. He graduated from the MIPT, in 2020. He is currently pursuing the Master of Science degree with the Skolkovo Institute of Science and Technology. He is also a Data Scientist at the NLP Department, Huawei, and works on adversarial attacks for machine translation.

EKATERINA DMITRIEVA is currently pursuing the Ph.D. degree with the CS Faculty, HSE University. Her research interests include semantic parsing, in particular text2SQL models and adversarial attacks.

NIKITA KLYUCHNIKOV received the M.Sc. degree in information science and technology from the Skolkovo Institute of Science and Technology, the M.Sc. degree in applied mathematics and physics from the Moscow Institute of Physics and Technology, in 2016, and the Ph.D. degree in computational and data science and engineering from the Skolkovo Institute of Science and Technology, in 2021. His main research interests include machine learning, Bayesian optimization, and their industrial applications.

ANDREY KRAVCHENKO is currently a Researcher at the University of Oxford and the Skolkovo Institute of Science and Technology. His Ph.D. research was at the intersection of machine learning and unstructured data extraction. He also played a significant role in the DIADEM project, which produced state-of-the art research in the field of large-scale fully automated web data extraction. His current research interests include the theory and application of anomaly detection in big data using sequences and graphs, and in particular, the development of efficient machine learning algorithms based on the embedding of vectors. He also works on exploring the broader connection between black-box machine learning models and knowledge-based systems, with a particular focus on knowledge graphs.

EKATERINA ARTEMOVA graduated from HSE University. She received the Ph.D. degree from the Institute of System Analysis, RAS. She is currently a Postdoctoral Researcher at the CS Faculty, HSE University, and advises the Noah Ark's NLP Team on advanced research topics. She focuses on NLU tasks, ranging from ToD systems to IE and creating new datasets.

EVGENIA KOMLEVA graduated from the MIPT, in 2021. She is currently pursuing the master's degree in data science with the Skolkovo Institute of Science and Technology. She is also working on NLP problems at ABBYY and plans to continue her research on adversarial attacks.

EVGENY BURNAEV received the M.Sc. degree from the Moscow Institute of Physics and Technology, in 2006, and the Ph.D. degree from the Institute for Information Transmission Problems, in 2008. He is currently an Associate Professor at the Skolkovo Institute of Science and Technology, Moscow, Russia. His research interests include Gaussian processes for multi-fidelity surrogate modeling and optimization, deep learning for 3D data analysis and manifold learning, and on-line learning for prediction and anomaly detection.