Gradient-based adversarial attacks on categorical sequence models via traversing an embedded world

Ivan Fursov[1], Alexey Zaytsev[1], Nikita Kluchnikov[1], Andrey Kravchenko[2], and Evgeny Burnaev[1]

[1] Skolkovo Institute of Science and Technology, Moscow, Russia
{ivan.fursov,a.zaytsev,n.kluchnikov,e.burnaev}@skoltech.ru
[2] DeepReason.ai, Oxford, UK,
andrey.kravchenko@deepreason.ai

Abstract. Deep learning models suffer from a phenomenon called adversarial attacks: we can apply minor changes to the model input to fool a classifier for a particular example. The literature mostly considers adversarial attacks on models with images and other structured inputs. However, the adversarial attacks for categorical sequences can also be harmful. Successful attacks for inputs in the form of categorical sequences should address the following challenges: **(1)** non-differentiability of the target function, **(2)** constraints on transformations of initial sequences, and **(3)** diversity of possible problems. We handle these challenges using two black-box adversarial attacks. The first approach adopts a Monte-Carlo method and allows usage in any scenario, the second approach uses a continuous relaxation of models and target metrics, and thus allows a usage of state-of-the-art methods for adversarial attacks with little additional effort. Results for money transactions, medical fraud, and NLP datasets suggest that the proposed methods generate reasonable adversarial sequences that are close to original ones, but fool machine learning models.

Keywords: Adversarial attack · Discrete Sequential data · Natural Language Processing.

1 Introduction

The deep learning revolution has led to the usage of deep neural network-based models across all sectors in the industry: from self-driving cars to oil and gas. However, the reliability of these solutions are questionable due to the vulnerability of almost all of the deep learning models to adversarial attacks [1] in computer vision [2,3], NLP [4,5], and graphs [6]. The idea of an adversarial attack is to modify an initial object, so the difference is undetectable to a human eye, but fools a target model: a model misclassifies the generated object, whilst for a human it is obvious that the class of the object remains the same [7].
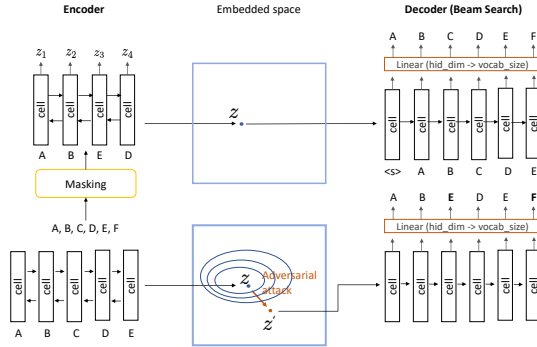
Fig. 1. Top figure: learning of our seq2seq model with the masking of tokens in an initial sequence. We also use beam search and an attention mechanism. Bottom figure: our adversarial attack, modification of a sequence $\mathbf{z}$ in the embedded state to be sure that the decoding of the adversarial sequence $D(\mathbf{z}')$ is close to the decoding $D(\mathbf{z})$, whilst the classifier score is significantly different.

For images we can calculate derivatives of the class probabilities with respect to the colour of pixels in an input image. Thus, moving along this direction we can apply slight alterations to a few pixels, and get a misclassified image, whilst keeping the image almost the same. For different problem statements attacks can be different, but in general a continuous space of images is rich enough for providing adversarial images.

The situation is different for sequential categorical data due to its discrete nature and thus absence of partial derivatives with respect to the input. The space of possible modifications is also limited. For certain problems a malicious user can not modify an object arbitrarily. For example, whilst trying to increase a credit score we can not remove a transaction from the history available to the bank; we only add another transaction. Both of these difficulties impose additional challenges for creation of adversarial attacks for categorical sequential data.

A survey on adversarial attacks for sequences [4,5] presents a list of possible options to overcome these difficulties. With respect to white-box attacks, there are two main research directions. Many approaches work with the initial space of tokens as input attempting to modify these sequences of tokens using operations like addition or replacement [8,9,10]. Another idea is to move into an embedded space and leverage on gradients-based approaches in this space [11]. We also note that most of these works focus on text sequence data.

We propose two approaches that can alleviate the aforementioned problems with differentiability and a limited space of modification actions, and work in the space of embedded sequences. The first approach is based on a Monte-Carlo search procedure in an embedded space, treating as the energy the weighted sum of the distance between the initial sequence and the generated one and the difference between the probability scores for them. The first term keeps two sequences

Table 1. Examples of adversarial sequences generated by the baseline HotFlip and our CASCADA approaches for the AG news dataset. HotFlip often selects the same strong word corrupting the sequence semantics and correctness. CASCADA is more ingenious and tries to keep the semantics, whilst sometimes changing the sequence too much.

Initial sequence x	HotFlip adversarial	CASCADA adversarial
jayasuriya hits back for sri lanka	jayasuriya arafat back for sri lanka	snow hits over back for sri lanka
determined jones jumps into finals	arafat jones jumps into finals	ibm music jumps into match
tiny memory card for mobiles launched	tiny memory card for economy economy	artificial memory card for hewitt pistons
nokia plots enterprise move	nokia plots economy economy	nokia steers enterprise move
sony shrinking the ps	sony shrinking economy ps	sony blames the indies cross
sackhappy d bags bills	google d bags bills	textile d bags bills
tunisian president ben ali reelected	nba president ben ali reelected	bayern hat toshiba got reelected

close to each other, whilst the second term identifies our intention to fool the classifier and generate a similar but misclassified example for a particular object. This approach is universal, as it does not require derivatives for the first and second terms whilst traversing the embedded space. The number of hyperparameters remains small, and each hyperparameter is interpretable with respect to the problem statement. The second approach illustrates adopts differentiable versions of sequential distance metrics. We use a trained differentiable version of the Levenshtein distance [12] and a surrogate classifier defined on embeddings of sequences. In this case our loss is differentiable, and we can adopt any gradient-based adversarial attack. The two approaches, which we name MCMC and CASCADA attacks, are summarised in Figure 1. Examples of generated sequences for the AG News dataset are presented in Table 1.

The generative model for adversarial attacks is a seq2seq model with masking [13]. So, the constructed RNN model can be reused for generating adversarial attacks based on these two approaches and creating adversarial attacks with a target direction as well as training embeddings for sequences. The validation of our approaches includes testing on diverse datasets from NLP, bank transactions, and medical insurance domains.

To sum up, we consider the problem of adversarial attack generation for categorical sequential data. The main contributions of this work are the following.

- Our first approach is based on an adaptation of Markov Chain Monte Carlo methods.
- Our second approach uses a continuous relaxation of the initial problem. This makes it possible to perform a classic gradient-based adversarial attack after applying a few new tricks.
- We construct seq2seq models to generate adversarial attacks using an attention mechanism and a beam search, and test the performance for attacking

models based on different principles, e.g. logistic regression for TF-IDF features from a diverse set of domains.

– Our adversarial attacks outperform the relevant baseline attacks; thus it is possible to construct effective attacks for categorical sequential data.

2   Related work

There exist adversarial attacks for different types of data. The most popular targets for adversarial attacks are images [14,15], although some work has also been done in areas such as graph data [16] and sequences [17].

It seems that one of the first articles on the generation of adversarial attacks for discrete sequences is [17]. The authors correctly identify the main challenges for adversarial attacks for discrete sequence models: a discrete space of possible objects and a complex definition of a semantically coherent sequence. Their approach considers a white-box adversarial attack with a binary classification problem. We focus on black-box adversarial attacks for sequences. This problem statement was considered in [18,9,19].

Extensive search among the space of possible sequences is computationally challenging [20], especially if the inference time for a neural network is significant. Authors of [18] identify certain pairs of tokens and then permute their positions within these pairs, thus working directly on a token level. Another black-box approach from [9] also performs a search at the token level.

It is also possible to use gradients for embeddings [11]. However, the authors of [11] limit directions of perturbations by moving towards another word in an embedded space, and the authors of [11,21] traverse the embedding space, whilst achieving limited success due to the outdated or complex categorical sequence models. Also, they consider only general perturbations and only NLP problems, whilst it is important to consider more general types of sequences.

As we see from the current state of the art, there is still a need to identify an effective end2end way to explore the space of categorical sequences for the problem of adversarial attacks generation. Moreover, as most of the applications focus on NLP-related tasks, there is still a room for improvement by widening the scope of application domains for adversarial attacks on categorical sequences. Among the methods presented in the literature we highlight HotFlip [10] as the most justified option, so we use compare it with our embeddings-based methods.

3   Methods

We start this section with the description of the general sequence-to-sequence model that we use to generate adversarial sequences, with some necessary details on model training and structure. We then describe the classifier model that we fool using our adversarial model. Next, we describe, how our seq2seq model is used to generate adversarial examples and present our MCMC and CASCADA adversarial attacks. Finally, we provide a description of how to obtain a differentiable version of the Levenshtein distance.

3.1   Models

Sequence-to-sequence models. Seq2seq models achieve remarkable results in various NLP problems, e.g. machine translation [22], text summarisation [23], and question answering [24]. These models have an encoder-decoder architecture: it maps an initial sequence $\mathbf{x}$ to dense representation using an encoder $\mathbf{z} = E(\mathbf{x})$ and then decodes it using a decoder $\mathbf{x}' = D(\mathbf{z})$ back to a sequence.

Following the ideas from CopyNet [25], we use a seq2seq model with an attention mechanism [22] for the copying problem and train an encoder and a decoder such that $\mathbf{x}' \approx \mathbf{x}$. The final network is not limited to copying the original sequence, but also discovers the nature of the data providing a language model. As the encoder $E(\mathbf{x})$ we use a bi-directional LSTM [26], and as the decoder $D(\mathbf{x})$ we use a uni-directional LSTM with Beam Search [27].

To train the model we mask some tokens from an input sequence, whilst trying to recover a complete output sequence, adopting ideas from MASS [28] and training a CopyNet [25] with the task to reconstruct an initial sequence. Masking techniques include swap of two random tokens, random deletion, random replacement by any other token, and random insertion. The objective for training the model is cross-entropy [29]. As we do not need any labelling, this unsupervised problem is easy to define and train.

In addition, we input a set of possible masking operations $\mathbf{m} = \{m_1, \ldots, m_s\}$. An example of such a set is $\mathbf{m} = \{AddToken, Replace, Delete\}$. We provide $\mathbf{m}$ to the model in addition to input sequence $\mathbf{x}$. As another example, for bank transactions, we can only use the addition of new tokens and $\mathbf{m} = \{AddToken\}$.

Classification models. As a classifier $C(\mathbf{x})$ we use a one-layer bi-directional LSTM with one fully-connected layer over the concatenation of the mean $\frac{1}{d}\sum_{i=1}^{d} z_i$ and $\max(\mathbf{z})$ of a hidden state $\mathbf{z} = \{z_1, \ldots, z_d\}$ or a logistic regression with TF-IDF features. A classifier takes a sequence $\mathbf{x}$ as input and outputs class probabilities (a classifier score) $C(\mathbf{x}) \in [0,1]^k$, where k is the number of classes or a class label $c(\mathbf{x})$ on the base of class probability scores $C(\mathbf{x})$.

3.2   Generation of adversarial sequences

We generate adversarial sequences for a sequence $\mathbf{x}$ by a targeted modification of a hidden representation $\mathbf{z} = E(\mathbf{x})$ given by encoder $E(\cdot)$ in such a way that the decoder generates an adversarial sequence $A(\mathbf{x})$ that is **(1)** similar to the original sequence and **(2)** have a lower probability of a targeted label.

The general attack scheme is presented in Algorithm 1. This attack works under the black-box settings: an attacker has no access to the targeted model. The algorithm uses an encoder, a decoder, word error rate WER between a generated and the initial sequences and a classifier that outputs class probability $C(\mathbf{x})$, and a class label $c(\mathbf{x})$. Slightly abusing the notation we refer to $C = C(\mathbf{x})$ as the classifier score for a class we want to attack in case of multiclass classification. CASCADA attack also uses a surrogate classifier and a surrogate word error rate distance.

The attack algorithm generates a set $\{\mathbf{z}_1, \ldots, \mathbf{z}_N\}$ of adversarial candidates via consecutive steps $\mathbf{z}_i := G(\mathbf{z}_{i-1})$ in the embedded space starting at $\mathbf{z}$ and selects the best one from the set. The difference between algorithms is in which function $G(\mathbf{z})$ we use.

Input: Number of steps N
Data: Original sequence $\mathbf{x}$
   and true label $c_{\mathbf{x}}$
Result: Adversarial
   sequence $\mathbf{x}^* = A(\mathbf{x})$
$\mathbf{z}_0 = E(\mathbf{x})$;
for $i \leftarrow 1$ **to** N **do**
   | % attack generator step;
   | $\mathbf{z}_i := G(\mathbf{z}_{i-1})$;
   | $C_i := C(D(\mathbf{z}))$ % score;
   | generate class label c_i
   | from score C_i;
   | $w_i = WER(D(\mathbf{z}_i), \mathbf{x})$;
end
if $\exists i$ *s.t.* $c_i \neq c_{\mathbf{x}}$ **then**
   | $\mathbf{x}^* = \mathbf{x}_i$ s.t.
   | $i = \arg\min_{i:c_i \neq c_{\mathbf{x}}} w_i$;
else
   | $\mathbf{x}^* = \mathbf{x}_i$ s.t.
   | $i = \arg\min_i C_i$;
end

Algorithm 1: The general attack scheme

Input: Embedding $\mathbf{z}$,
   proposal variance σ^2,
   energy temperatures
   $\sigma_{wer}, \sigma_{class}$, initial
   class label c_0
Result: Attacked embedding
   $\mathbf{z}' = G(\mathbf{z})$
$\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 I)$;
$\mathbf{z}' := \mathbf{z} + \boldsymbol{\varepsilon}$;
$\mathbf{x}' := D(\mathbf{z}')$;
$C := C(\mathbf{x}')$;
generate class label c from
   score C;
$w = WER(\mathbf{x}', \mathbf{x})$;
$\alpha = \exp\left(\frac{-w}{\sigma_{wer}} + \frac{-[c_0 = c]}{\sigma_{class}}\right)$;
$u \sim \mathcal{U}([0, 1])$;
if $\alpha < u$ **then**
   | $\mathbf{z}' := \mathbf{z}$;
end

Algorithm 2: The MCMC attack defines a generator step $\mathbf{z}_i := G(\mathbf{z}_{i-1})$, $[\cdot]$ is the indicator function

Naïve random walk attack. The natural approach for generating a new sequence $\mathbf{x}^*$ in an embedded space is a random jump to a point $\mathbf{z}^*$ in that embedded space from the embedding of an initial sequence $\mathbf{z} = E(\mathbf{x})$. An adversarial candidate is a decoder output $\mathbf{x}^* = D(\mathbf{z}^*)$. As we have a total budget N, we make up to N steps until we find a sufficiently good sequence. Whilst this algorithm seems to be quite simple, it can provide a good baseline against more sophisticated approaches, and can work well enough for an adequate embedding space.

Formally, for this variation of Algorithm 1 we use $\mathbf{z}' = G(\mathbf{z}) = \mathbf{z} + \boldsymbol{\varepsilon}, \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 I)$ with σ^2 being a hyperparameter of our algorithm. Note that in the case of a random walk we defer from the general attack scheme, and each time use the same initial sequence $\mathbf{z}_0 = E(\mathbf{x})$ instead of $\mathbf{z}_{i-1}$ to get a new sequence $\mathbf{z}_i$.

MCMC walk. Markov chain Monte Carlo (MCMC) can lead to a more effective approach. We generate a new point using Algorithm 1 with $G(\cdot)$ defined in Algorithm 2 by an MCMC walk. This walk takes into account the similar-

ity between the initial and the generated sequences and the adversity of the target sequence, so we can generate point $\mathbf{z}_i := G(\mathbf{z}_{i-1})$ at each step more effectively. Similar to the naïve random walk, the MCMC uses the noise variance for embedded space σ. In addition, the MCMC walk approach has temperature parameters σ_{wer} and σ_{class} that identify the scale of the energy we are seeking, and what is the trade-off between the distance among sequences and the drop in the classification score.

The MCMC random walk is designed to make smarter steps and traverses through the embedded space.

CASCADA attack. Naïve and MCMC attacks can be inefficient. Both of these approaches are computationally expensive for deep seq2seq architectures.

The CASCADA (CAtegorical Sequences Continuous ADversarial Attack) attack is an end-to-end approach, which computes the WER metric and runs a beam search only once.

In the CASCADA approach we use Deep Levenshtein model $WER_{deep}(\mathbf{z}, \mathbf{z}')$ [12] and a surrogate classification model $C_s(\mathbf{z})$ on top of a seq2seq CopyNet. Both of these models act in the embeddings space. Therefore, we can evaluate derivatives with respect to arguments of $WER_{deep}(\mathbf{z}_0, \mathbf{z})$ and $C_s(\mathbf{z})$ inside the target function, thus making it possible to run a gradient-based optimisation that tries to select the adversarial sequence with the best score.

We search for a minimum of a function $C_s(\mathbf{z}) + \lambda WER_{deep}(\mathbf{z}, \mathbf{z}_0)$ with respect to $\mathbf{z}$. The hyperparameter λ identifies a trade-off between trying to get a lower score for a classifier and minimising the distance between $\mathbf{z}$ and the initial sequence $\mathbf{z}_0$. So, the attack $\mathbf{z}'$ is a solution of the optimisation problem:

$$\mathbf{z}' = \arg\min_{\mathbf{z}} C_s(\mathbf{z}) + \lambda WER_{deep}(\mathbf{z}, \mathbf{z}_0).$$

After the generation of a set of candidates during the gradient descent optimisation $\mathbf{z}_1, \ldots, \mathbf{z}_N$, we apply the decoder to each candidate, obtaining $\mathbf{x}_1 = D(\mathbf{z}_1), \ldots, \mathbf{x}_N = D(\mathbf{z}_N)$ as a set of adversarial candidates.

Deep Levenshtein. To make gradient-based updates to an embedded state, we use a differentiable version of the Levenshtein distance function [30]. We use the Deep Levenshtein distance proposed by [12] and considered also in [30]. In our case, WER is used instead of the Levenshtein distance, since we work on the word level instead of the character level for NLP tasks, and for non-textual tasks there are simply no levels other than "token" level.

To collect the training data for each dataset we generate about 2 million pairs. For each pair we apply masks similar to CopyNet, obtaining an original sequence and a close but different sequence. We have also added pairs composed of different sequences from the training data for a better coverage of distant sequences. Our target is $WER_{norm}(\mathbf{x}, \mathbf{y}) = \frac{WER(\mathbf{x}, \mathbf{y})}{\max(|\mathbf{x}|, |\mathbf{y}|)}$. We train a model $M(\mathbf{z})$ with the objective $\|\frac{1}{2}(\cos(M(E(\mathbf{x})), M(E(\mathbf{y}))) + 1) - WER_{norm}(\mathbf{x}, \mathbf{y})\|$. The mean absolute error for the learned Deep Levenstein distance $WER_{deep}(\mathbf{z}, \mathbf{z}') = \frac{1}{2}(\cos(M(\mathbf{z}), M(\mathbf{z}')) + 1)$ is 0.15 for all considered datasets.

4   Experiments

In this section we describe our experiments. The datasets and the source code are published online[3].

4.1   Datasets

To test the proposed approaches we use NLP, bank transactions, and medical sequence datasets.

We use **NLP dataset** AG news [31] dedicated to topic identification. The four largest classes from the corpus constitute our dataset. The number of training samples for each class is $30,000$ and the number of test samples is $1,900$. We also use **a transactions dataset**, aimed at predicting gender [4]. We use sequences of transactions codes (gas station, art gallery, etc.) and transaction amounts as an input. We also supplement these datasets with another dataset from the **medical insurance** [20] domain. The goal is to detect frauds based on a history of visits of patients to a doctor. Each sequence consists of visits with information about a drug code and amount of money spent for each visit.

For the attacked logistic regression model with TF-IDF features as inputs, the macro-average ROC AUC scores for Transcations-GENDER, Healthcare Insurance and AG News datasets are 0.70, 0.74, 0.88, and 0.96 correspondingly.

Preprocessing of the datasets. For AG news we use a standard preprocessing procedure. For the healthcare insurance dataset each sequence of tokens consists of medical codes or the procedure assigned after the next visit to a clinic, and a label if the entire sequence for a patient is a fraud or not, with the percentage of frauds in the available dataset being 1.5% and total number of patients being $381,013$.

For the transactions datasets the preprocessing is more complex, so we describe it separately. For the gender prediction dataset we compose each token from the transaction type, the Merchant Category Code (MCC), and the transaction amount bin. We split all amounts into decile bins and then sort them, so index 0 corresponds to the cheapest purchases and index 9 corresponds to the most expensive purchases. An example encoding of a token from a sequence of transactions is 4814_1030_3 with 4814 being the MCC code, 1030 being the transaction type and 3 the index of the decile amount bin. Each sequence corresponds to transactions during the last three days with the mean sequence length being 10.25.

4.2   Metrics

The two types of metrics for the evaluation of the quality of adversarial attacks on sequences are the difference in the classifier score between an initial and a generated adversarial sequences and the distance between these sequences.

---

[3] The code is available at
https://github.com/fursovia/dilma/tree/master. The data is available at
https://www.dropbox.com/s/axu26guw2a0mwos/adat_datasets.zip?dl=0.

[4] https://www.kaggle.com/c/python-and-analyze-data-final-project/data

To measure the performance of the proposed approaches we use three metrics that identify the accuracy drop after adversarial attacks: the ROC AUC drop, the accuracy drop, and the mean classifier score drop. To measure the difference for the new adversarial sequences we use the word error rate (WER) between the initial and generated adversarial sequences.

We also propose a new metric for evaluating adversarial attacks on classifiers for categorical sequences, which combines distance-based and score-based approaches. To get a more realistic metric we perform a normalisation using WERs between the initial and adversarial sequences, which we call the normalised accuracy drop $\text{NAD}(A) = \frac{1}{|Z|} \sum_{i \in Z} 1\{c(\mathbf{x}_i) \neq c(A(\mathbf{x}_i))\} \left(\frac{L_i - \text{WER}(A(\mathbf{x}_i), \mathbf{x}_i)}{L_i - 1} \right)$, where $c(\mathbf{x})$ outputs class labels instead of probabilities $C(\mathbf{x})$, $Z = \{i | c(\mathbf{x}_i) = y_i\}$, and L_i is the maximum length of $\mathbf{x}_i$ and the adversarial sequence $\mathbf{x}'_i = A(\mathbf{x}_i)$ generated by the adversarial attack A.

4.3   Main experiment for adversarial attacks

We compare our approach with the current state of the art, HotFlip [10]. HotFlip at each step selects the best token to change, given an approximation of partial derivatives for all tokens and all elements of the dictionary. To complete the HotFlip attack in our setting we generate N sequences with beam search and then follow our general selection procedure described in Algorithm 1.

We run experiments to keep WER similar for the four considered approaches: HotFlip, random walk attack, MCMC walk attack, and CASCADA. We select hyperparameters to get approximately similar WER scores for different approaches. We generate $N = 100$ sequences for each of the four approaches and select the best one according to the criterion described above.

In Table 2 we present results for the proposed approaches, whilst attacking an independent logistic regression model with TF-IDF features and using LSTM model as a surrogate classifier. We see that embedding-based approaches provide decent performance and are a better way to generated more adversarial examples, while NAD metric puts too significant emphasis on WER values when comparing different approaches.

4.4   Constrained adversarial attack

We compare the performance of general and constrained adversarial attacks. In the first case the attack applies all possible modifications to sequences. In the second case only certain perturbations are allowed, e.g. an addition of a token or swapping two tokens. The comparison of performances for various attacks is presented in Table 3: all types of attacks have comparable performances for our CASCADA approach.

4.5   Reliability study

The selection of hyperparameters often affects the performance of an adversarial attack. We run 599 different hyperparameters configurations for training seq2seq

Table 2. Fooling logistic regression with TF-IDF representations as inputs by running the considered attacks on the four diverse datasets. We maximise metrics with the ↑ signs and minimise metrics with the ↓ signs. Embedding-based methods work better when looking both at perplexity and accuracy drops.

Transactions Gender	ROC AUC drop ↑	Accuracy drop ↑	Probability drop ↑	Normalised WER ↓	Log perplexity ↓	NAD ↑
Random walk	0.539	0.40	0.189	0.561	4.29	0.334
HotFlip	0.243	0.26	0.091	**0.100**	5.15	**0.623**
MCMC walk	**0.640**	**0.55**	**0.245**	0.719	**4.28**	0.333
CASCADA	0.361	0.32	0.121	0.198	4.49	0.426
AG News						
Random walk	0.406	0.66	0.487	0.704	5.21	0.274
HotFlip	0.342	0.67	0.477	**0.218**	6.76	**0.723**
MCMC walk	**0.452**	**0.72**	**0.525**	0.757	**5.16**	0.270
CASCADA	0.422	0.62	0.492	0.385	6.29	0.494
Healthcare insurance						
Random walk	0.566	0.47	0.094	0.725	4.90	0.258
HotFlip	**0.778**	**0.92**	**0.294**	**0.464**	6.75	**0.371**
MCMC walk	0.364	0.29	0.062	0.695	4.50	0.194
CASCADA	0.131	0.26	0.045	0.492	**4.28**	0.106

Table 3. Constrained adversarial attacks on logistic regression with TF-IDF using various masking tokens for the AG news dataset. Log perplexity is almost similar for all approaches.

Masker	Accuracy drop ↑	Normalised WER ↓	NAD ↑
No constraints	**0.62**	**0.39**	**0.492**
Add	**0.62**	0.51	0.382
Replace	0.59	0.50	0.366
Swap	0.61	0.52	0.333

models, trained with attention and masking, and the CASCADA adversarial attack based on this model. The results are presented in Figure 2. We observe that by varying hyperparameters, we select a trade-off between the similarity of initial sequence and an adversarial one and corresponding classifier probability drop. Moreover, varying of hyperparameters for a selected trade-off we observe robust results without significant drop of quality for particular runs or particular configurations.

5   Conclusion

A construction of an adversarial attack for a categorical sequence is a challenging problem. We consider two approaches to solve this problem: directed random modifications and two differentiable surrogates, for a distance between sequences and for a classifier, that act from an embedded space. The first approach is based on the application of MCMC to generated sequences, and the second
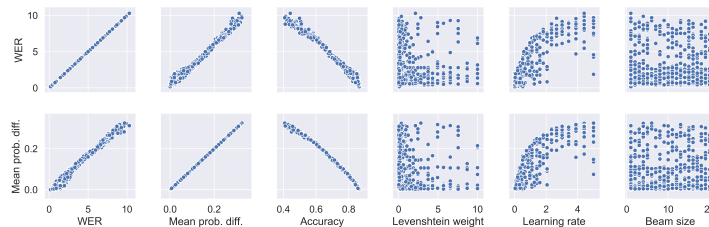
Fig. 2. Mean WER and accuracy drops for various configurations of hyperparameters for the Transactions Gender dataset: the learning rate, the Deep Levenshtein weight, and the beam number. Mean WER and accuracy drop are inversely related as expected, whilst the seq2seq model is robust against changes of hyperparameter values.

approach uses surrogates for constructing gradient attacks. At the core of our approaches lies a modern seq2seq architecture, which demonstrates an adequate performance. To improve results we adopt recent ideas from the NLP world, including masked training and the attention mechanism.

For considered applications, which include NLP, bank card transactions, and healthcare, our approaches show a reasonable performance with respect to common metrics for adversarial attacks and sequence distances. Moreover, we can limit the space of possible modifications, e.g. use only addition operations during an adversarial sequence generation.

Acknowledgments

References

1. X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 9, pp. 2805–2824, 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. V. Khrulkov and I. Oseledets, "Art of singular vectors and universal adversarial perturbations," in *IEEE CVPR*, pp. 8562–8570, 2018.
4. W. E. Zhang, Q. Z. Sheng, A. Alhazmi, and C. Li, "Adversarial attacks on deep-learning models in natural language processing: A survey," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 11, no. 3, pp. 1–41, 2020.
5. W. Wang, B. Tang, R. Wang, L. Wang, and A. Ye, "A survey on adversarial attacks and defenses in text," *arXiv:1902.07285 preprint*, 2019.
6. L. Sun, J. Wang, P. S. Yu, and B. Li, "Adversarial attack and defense on graph data: A survey," *arXiv:1812.10528 preprint*, 2018.
7. A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," in *ICLR*, 2017.
8. S. Samanta and S. Mehta, "Towards crafting text adversarial samples," *arXiv:1707.02812 preprint*, 2017.

9. B. Liang, H. Li, M. Su, P. Bian, X. Li, and W. Shi, "Deep text classification can be fooled," in *IJCAI*, 2017.
10. J. Ebrahimi, A. Rao, D. Lowd, and D. Dou, "Hotflip: White-box adversarial examples for text classification," in *Annual Meeting of ACL*, pp. 31–36, 2018.
11. M. Sato, J. Suzuki, H. Shindo, and Y. Matsumoto, "Interpretable adversarial perturbation in input embedding space for text," in *IJCAI*, 2018.
12. S. Moon, L. Neves, and V. Carvalho, "Multimodal named entity recognition for short social media posts," in *Conference of the North American Chapter ACL: Human Language Technologies*, pp. 852–860, 2018.
13. S. Bowman, L. Vilnis, O. Vinyals, A. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space," in *SIGNLL CoNNL*, pp. 10–21, 2016.
14. C. Szegedy, W. Zaremba, I. Sutskever, J. B. Estrach, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *ICLR*, 2014.
15. I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *ICLR*, 2014.
16. D. Zügner, A. Akbarnejad, and S. Günnemann, "Adversarial attacks on neural networks for graph data," in *ACM SIGKDD*, pp. 2847–2856, 2018.
17. N. Papernot, P. McDaniel, A. Swami, and R. Harang, "Crafting adversarial input sequences for recurrent neural networks," in *IEEE MILCOM*, pp. 49–54, 2016.
18. J. Gao, J. Lanchantin, M. L. Soffa, and Y. Qi, "Black-box generation of adversarial text sequences to evade deep learning classifiers," in *IEEE Security and Privacy Workshops*, pp. 50–56, IEEE, 2018.
19. 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 *AAAI*, 2020.
20. I. Fursov, A. Zaytsev, R. Khasyanov, M. Spindler, and E. Burnaev, "Sequence embeddings help to identify fraudulent cases in healthcare insurance," *arXiv:1910.03072 preprint*, 2019.
21. 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," *arXiv preprint arXiv:2003.10388*, 2020.
22. D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *ICLR*, 2015.
23. P. Li, W. Lam, L. Bing, and Z. Wang, "Deep recurrent generative decoder for abstractive text summarization," in *EMNLP*, pp. 2091–2100, 2017.
24. R. Hu, J. Andreas, and M. e. a. Rohrbach, "Learning to reason: End-to-end module networks for visual question answering," in *IEEE ICCV*, pp. 804–813, 2017.
25. J. Gu, Z. Lu, H. Li, and V. O. Li, "Incorporating copying mechanism in sequence-to-sequence learning," in *Annual meeting of ACL*, pp. 1631–1640, 2016.
26. F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," 1999.
27. A. Graves, "Sequence transduction with recurrent neural networks," *arXiv:1211.3711 preprint*, 2012.
28. K. Song, X. Tan, T. Qin, J. Lu, and T.-Y. Liu, "Mass: Masked sequence to sequence pre-training for language generation," in *ICML*, pp. 5926–5936, 2019.
29. K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: a method for automatic evaluation of machine translation," in *Annual meeting of ACL*, pp. 311–318, 2002.
30. I. Fursov, A. Zaytsev, and et. al., "Differentiable language model adversarial attacks on categorical sequence classifiers," *arXiv:2006.11078 preprint*, 2020.
31. X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *NeurIPS*, pp. 649–657, 2015.