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

Ivan Fursov  
Skoltech  
Moscow, Russia

Alexey Zaytsev  
Skoltech  
Moscow, Russia

Nikita Kluchnikov  
Skoltech  
Moscow, Russia

Andrey Kravchenko  
DeepReason.ai  
Oxford, UK

Evgeny Burnaev  
Skoltech  
Moscow, Russia

Abstract

An adversarial attack paradigm explores various scenarios for vulnerability of machine and especially deep learning models: we can apply minor changes to the model input to force a classifier’s failure for a particular example. Most of the state of the art frameworks focus on adversarial attacks for images and other structured model inputs. The adversarial attacks for categorical sequences can also be harmful if they are successful. However, successful attacks for inputs based on categorical sequences 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 approaches. The first approach adopts Monte-Carlo methods and allows usage in any scenario, the second approach uses a continuous relaxation of models and target metrics, and thus allows using general state of the art methods on adversarial attacks with little additional effort. Results for money transactions, medical fraud, and NLP datasets suggest the proposed methods generate reasonable adversarial sequences that are close to original ones, but fool machine learning models even for blackbox adversarial attacks.

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 [1] to oil and gas [2]. However, the reliability of these solutions are questionable due to the vulnerability of almost all of the deep learning models to adversarial attacks [3] including models in computer vision [4, 5], NLP [6, 7], and graphs [8]. The idea of an adversarial attack is to generate an object that fools a deep learning model by inserting changes into the initial object, undetectable to a human eye: a deep learning model misclassifies the generated object, whilst for a human it is obvious that the class of the object remains the same [9].

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

A survey on adversarial attacks for sequences [6, 7] presents a list of possible options to overcome these difficulties. With respect to white-box attacks, there are two main research directions at the moment. Many approaches that accept the initial space of tokens as input attempt to modify these sequences using operations like addition, replacement, or switching of tokens [10, 11, 12]. Another idea is to move into an embedded space and leverage on gradients and optimisation approaches in that space [13]. 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 mostly in the space of embedded sequences. The first approach is based on 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 score for these sequences. The first term keeps two sequences 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 that we can adopt differentiable versions of sequential distance metrics. We use a trained differentiable version of the Levenshtein distance [14]. 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 AD News dataset are presented in Table[1].

As a generative model for adversarial attacks we use a seq2seq model with masking [15], thus allowing the constructed RNN model to be reused for generating adversarial attacks based on these two approaches and create 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, the main contributions of this work are the following.

- We consider the problem of adversarial attack generation for categorical sequential data.
- Our first approach is based on an adaptation of 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 [16, 17], although some work has also been done in areas such as graph data [18] and sequences [19].

It seems that one of the first sources for generation of adversarial attacks for discrete sequences is [19]. 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 focuses on a white-box adversarial attack with a binary classification problem. In our work we also consider this problem statement, but focus on black-box adversarial attacks for sequences and target vs non-target attacks. This problem statement was considered in [20, 21]. Authors of [20] 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 [11] also performs a direct search for the most harmful tokens, trying to fool a classifier.

As stated in Section[1] the first research direction is a direct generation of sequences by trying to apply natural
| 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 mach |
| tiny memory card for mobiles launched | tiny memory card for | artificial memory card for |
| 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 |

Table 1: Examples of generated adversarial sequences for the AG news dataset evaluated on the baseline HotFlip and our CASCADA approaches. HotFlip often selects the same strong word corrupting the semantics of a sequence. CASCADA is more ingenious, whilst sometimes trying to change the sequence too much.

modifications like deletion or replacement of a token in a sequence, with the expectation that will result in a sufficiently good adversarial outcome. [10]. This idea leads to an extensive search among the space of possible sequences, thus making the problem computationally challenging, especially if the inference time for a neural network is significant [22]. Moreover, we have little control on the proximity of the initial and modified sequences, as we do not take into account the proximity of tokens in the embedded space.

The second research direction is closer to the common approaches used by adversarial attacks practitioners, as we try to use gradients of the cost function and distance measure [13]. However, the authors in [13] limit directions of perturbations by moving towards another word in an embedded space, which seems restrictive for state of the art RNNs that adopt mechanisms such as attention [23] and beam search [24] to increase the performance of the sequence model.

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

Markov chain Monte Carlo (MCMC) approaches provide a good candidate set of solutions for exploration of embedded spaces [25]. We select an energy function to reflect our views on criteria for good and bad sequences. For, example traversing complex spaces via MCMC is possible for the space of deep neural network parameters [26].

Another option is to define a differentiable loss function that takes into account the distance between the initial and generated sequences and simultaneously tries to minimise the class score for the generated adversarial example. Authors of [27] compose a white-box adversarial attack based on this idea, but due to a high divergence between the embedded and categorical sequence spaces, they return to the categorical sequence space at each step, thus making the search ineffective and not end to end.

For the class score it is relatively easy to define a differentiable version of it, whilst for the distance measure this problem is more complex. However, there are differentiable versions of distance measures, e.g. a differentiable BLEU score [28], a learned [14] or soft [29] Levenshtein edit distance.

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

3 Methods

This section we start 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 proceed with the description of how our seq2seq model is used to generate adversarial examples. Finally, we provide a description of how to obtain a differentiable version of the Levenshtein distance and how to adopt an MCMC approach to our problem.

3.1 Models

Sequence-to-sequence model Seq2seq models have achieved remarkable results in various NLP problems. In particular, they are commonly used for machine translation [30], text summarisation [31], dialogue sys-
Seq2seq models belong to an encoder-decoder like architecture, which maps an initial sequence $x$ to dense representations using encoder $z = E(x)$ and then decodes them using decoder $D(z)$ back to a sequence. For machine translation, an encoder transforms a text in one language to a hidden representation, and a decoder uses this representation to generate a sequence in another language. For text summarisation a larger text is replaced by a shorter one.

To train the model we mask some tokens from an input sequence trying to recover a complete output sequence, adopting ideas from MASS \[34\] and training a CopyNet \[35\] with the task to reconstruct an initial sequence. Masking techniques include among others swap of two random tokens, random deletion, random replacement by any other token, and random insertion. The final network is not limited to copying the original sequence, but also discovers the nature of the data. Thus, the model always doubts whether it needs to replace, delete, or insert tokens from the initial sequence to generate a better sequence. We define the mask procedure as a set of possible masking operations $M = \{m_1, \ldots, m_s\}$, with $m_i$. An example of such a set is \{AddRandomToken, Replace, Delete\}.

During the decoding stage, we can also use an attention mechanism for masks.

The objective for training the model is cross-entropy and the metric used for this objective is BLEU \[36\]. As we do not need any labelling, this unsupervised problem is easy to define and train.

Following the ideas from CopyNet \[35\], we use a seq2seq model with an attention mechanism \[30\] for copying. As the encoder we use a bi-directional LSTM \[37\], and as the decoder we use a uni-directional LSTM with Beam Search (see e.g. \[38\]).

**Classification model** In all experiments as a classifier $C(x)$ we use a one-layer bi-directional LSTM with one fully-connected layer over the concatenation of the mean $\frac{1}{d}\sum_{t=1}^{d} z_t$ and max of a hidden state $z = \{z_1, \ldots, z_d\}$. A classifier takes a sequence $x$ as input and outputs a class probability or a classifier score $C(x)$ which ranges from 0 to 1, as we consider here only binary classification problems, and a class label $c(x)$ on the base of a threshold $C_0$ and a probability score $C(x)$.

### 3.2 Adversarial sequence generation

The three approaches we propose for adversarial sequence generation are all based on the idea of modifying the hidden representation $z = E(x)$ of the encoded sequence $x$ in such a way that the decoder will generate an adversarial sequence $A(x)$ that will be (1) similar to the original sequence and (2) have a lower probability of a targeted label.

Attacks work under black-box settings: an attacker has no access to the targeted model.

The general attack scheme is presented in Algorithm \[1\]. The algorithm uses an encoder, a decoder, and a black-box classifier that outputs class probabilities $C(x)$ and a class label $c(x)$. The attack $G(z)$ is a function that acts in an embedded space to create a diverse set of adversarial candidates.

**Input:** Number of steps $N$

**Data:** Original sequence $x$ and true label $c_x$

**Result:** Adversarial sequence $x^* = A(x)$

1. $z_0 = E(x)$;
2. for $i \leftarrow 1$ to $N$ do
   1. $z_i := G(z_{i-1})$;
   2. $C_i = C(D(z_i))$;
   3. $c_i$ is class label on the base of score $C_i$;
   4. $w_i = WER(D(z_i), x)$;
3. end
4. if $\exists i$ s.t. $c_i \neq c_x$ then
   1. $x^* = x_i$ s.t. $i = \arg\min_{i:c_i \neq c_x} w_i$;
5. else
6. $x^* = x_i$ s.t. $i = \arg\min_{c_i}$;
7. end

**Algorithm 1:** The general attack scheme

#### 3.2.1 Nave random walk attack

The natural approach to generate a new sequence $x^*_i$ in an embedded space is to jump to another point $z^*_i$ in that embedded space from the embedding of a sequence $z_i = D(x_i)$ and get a new sequence $x^*_i = D(z^*_i)$ after an application of the decoder to that sequence. As we have a total budget $N$, we make up to $N$ steps until we find a sufficiently good sequence.

The formal attack algorithm $G(z)$ based on this idea is described in Algorithm \[2\]. Note that in the case of a random walk we defer from the general attack scheme, and each time use the same initial sequence $z_0$ to get a new sequence $z_i$ instead of $z_{i-1}$.

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 em-
bedding space.

**Input:** Embedding $z$, noise $\sigma^2$

**Result:** Attacked embedding $z' = G(z)$

$\varepsilon \sim N(0, \sigma^2)$; $z' := z + \varepsilon$.

**Algorithm 2:** The naive random walk attack

**Input:** Embedding $z$, noise $\sigma^2$

**Result:** Attacked embedding $z' = G(z)$

$z' := \arg\min z' \in \mathcal{G} C(z') + \lambda WER_{deep}(z', z)$ with the initial point $z$;

**Algorithm 3:** The CASCADA random walk attack

### 3.2.2 MCMC walk

Markov chain Monte Carlo (MCMC) can provide a more effective approach. We generate a new point using an MCMC walk that takes into account the similarity between the initial and the generated sequences and the adversity of the target sequence with the additional substitute model $c(z)$. The algorithm is presented in Algorithm 4. Note that the MCMC walk and random walk approaches share parameters: the number of possible steps $N$ and the noise variance for embedded space $\sigma$. In addition, the MCMC walk approach has temperature parameters $\sigma_{\text{weeks}}$ and $\sigma_{\text{class}}$, that identify the scale of the energy we are seeking for, and what is the trade-off between the distance among sequences and the drop in the classification score.

Our hope is that the MCMC version of walking makes smarter steps and traverses through embedded spaces in a more meaningful way. We observe classic convergence guarantees for MCMC.

**Input:** Embedding $z$, proposal variance $\sigma^2$, energy temperatures $\sigma_{\text{weeks}}, \sigma_{\text{class}}$, initial class label $c_0$

**Result:** Attacked embedding $z' = G(z)$

$\varepsilon \sim N(0, \sigma^2)$; $z' := z + \varepsilon$; $C = C(z')$; $c$ is a class label based on the score $C$; $w = WER(z', z)$; $\alpha = \exp \left( \frac{-w}{\sigma_{\text{weeks}}} + \frac{1+c_0-c}{\sigma_{\text{class}}} \right)$; $u \sim U([0, 1])$;

if $\alpha < u$ then

$z' := z$; $c := C(z')$; $w := WER(z', z)$;

end

**Algorithm 4:** The MCMC attack

### 3.2.3 HotFlip

The main idea of HotFlip [12] is to select the best token to change, given an approximation of partial derivatives for all tokens and all elements of the dictionary. As we can change many tokens, we can do so in a greedy way or by beam searching for a good sequence of changing tokens. To complete the HotFlip attack in our setting we generate $N$ sequences with beam search and then follow our general attack procedure described in [1].

### 3.2.4 CASCADA attack

Nave and MCMC attacks can be inefficient and slow. Each step requires a beam search run and a calculation of WER. Both of these approaches are computationally expensive for deep seq2seq architectures.

In the CASCADA approach we use Deep Levenshtein [14] and classification models on top of a seq2seq Copy-Net. Both of these models act in the space of embeddings and learn $WER_{deep}(z, z')$ and $C(z)$ correspondingly. Therefore, we can evaluate derivatives with respect to components of $WER_{deep}(z_0, z)$ and $C(z)$ inside the target function, thus making it possible to run a gradient-based optimisation trying to select the model with the best score.

During optimisation we search for a minimum of a function $C(z) + \lambda WER_{deep}(z, z_0)$ with respect to $z$. The hyperparameter $\lambda$ identifies a trade-off between trying to get a lower score for a classifier and minimising the distance between $z$ and the initial sequence $z_0$.

After the generation of a set of candidates during the gradient descent optimisation $z_1, \ldots, z_N$, we apply the decoder to each candidate, obtaining $x_1 = D(z_1), \ldots, x_N = D(z_N)$ as a set of adversarial candidates.

All steps of the CASCADA-based generation of a single sequence are summarised in the algorithm [3].

**Deep Levenshtein** To make gradient-based updates to an embedded state, we use a differentiable version of there Levenshtein distance function. There have been attempts to derive smooth approximations of edit distances [29].

However, the beam search operation is not differentiable and has a high computational complexity. To avoid this, we use the Deep Levenshtein distance proposed by [14]. In our case, WER is used instead of the Levenshtein dist-
tance, since we work at the word level instead of the character level for NLP tasks, and for non-textual tasks there are simply no levels other than “token”.

To collect the training data we have used examples generated from each dataset, around 2 million examples in total. For each example we have applied masks similar to the ones used in CopyNet. As a result, we have acquired pairs: each pair being a sequence and a close but different sequence obtained after the application of masking. We have also added pairs obtained from the training data to get a better coverage of far-away sequences. For each pair we calculated normalised $WER_{norm}(x, y) = \frac{WER(x, y)}{\max(|x|, |y|)}$ and similarity $\text{sim}(x, y) = 1 - WER_{norm}(x, y)$ metrics and learned a shared-bidirectional LSTM model $M(x_i)$ with the objective $\frac{1}{2}(\cos(M(x), M(y)) + 1) - \text{sim}(x, y)\|$. To improve the quality of the model, we have used concatenations of $M(x)$ and corresponding attention vectors of $M(y)$.

3.3 Targeted permutations

Another feature of our approach is targeted transformations of an attacked sequence. For the random search, MCMC search, and CASCADA we can pass desired transformations $m = \{m_1, \ldots, m_k\}$ as additional input during decoding. Thus, we are able to limit the type of transformations that can be applied during an adversarial attack. For example, for bank transactions, we can only use the addition of new tokens, as this can be the only possible way to modify a sequence of transactions, or add new tokens to the end of a sequence.

4 Experiments

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

4.1 Datasets

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

We use a NLP dataset AG news [39] dedicated to topic identification. The four largest classes from the corpus constitute our dataset. The number of training samples for each class is 30000 and the number of test samples is 1900. There are two open transactions datasets that we use in our work, aimed at predicting age [40] and gender [41]. We use sequences of transactions codes (gas station, art gallery, and so on) and transaction amounts as input for both datasets. We also supplement these datasets with another dataset from the medical insurance [22] 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.

4.1.1 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 381013. For the transactions datasets the preprocessing is more complex, so we describe it separately.

Transactions datasets For the gender prediction dataset we compose each token from the transaction type, the Merchant Category Code (MCC) code, and the transaction amount bin. We split all amounts into 10 equal bins and then sort them, so index 0 corresponds to very cheap purchases and index 9 corresponds to the most expensive ones. An example encoding of a token from a sequence of transaction 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.

For the age prediction dataset we proceed in a similar way, but use only the transaction type instead of the MCC code and the transaction type. An example encoding of a token from such this sequence is 35 7 with 35 being the transaction type and 7 the index of the decile amount bin.

4.2 Metrics

The two types of metrics for 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.

To measure the performance of 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 between the initial and generated adversarial sequences.

We also propose a new metric for evaluating adversar-
Table 2: Fooling the LSTM model by running four considered methods on four evaluation datasets. We maximise metrics with ↑ and minimise metrics with ↓ signs. We see, that CASCADA works better when looking both at perplexity and accuracy drops.

| Transactions, Age | ROC AUC drop ↑ | Accuracy drop ↑ | Probability drop ↑ | Normalised WER ↓ | Log perplexity ↓ | NAD ↑ |
|-------------------|----------------|-----------------|-------------------|----------------|-----------------|------|
| Random search     | 0.318          | 0.38            | 0.143             | 0.413          | 3.79            | 0.553 |
| HotFlip           | 0.264          | 0.14            | 0.125             | 0.104          | 4.83            | 0.926 |
| MCMC              | 0.114          | 0.15            | 0.064             | 0.158          | 3.85            | 0.122 |
| CASCADA           | 0.308          | 0.35            | 0.139             | 0.161          | 4.05            | 0.662 |

Table 3: Fooling logistic regression with TF-IDF representations as inputs by running four considered methods on four evaluation datasets. We maximise metrics with ↑ signs and minimise metrics with ↓ signs. We observe that CASCADA works better when looking both at perplexity and accuracy drops.

| Transactions, Age | ROC AUC drop ↑ | Accuracy drop ↑ | Probability drop ↑ | Normalised WER ↓ | Log perplexity ↓ | NAD ↑ |
|-------------------|----------------|-----------------|-------------------|----------------|-----------------|------|
| Random search     | 0.700          | 0.52            | 0.246             | 0.561          | 4.29            | 0.427 |
| HotFlip           | 0.406          | 0.35            | 0.246             | 0.100          | 5.15            | 0.984 |
| MCMC              | 0.715          | 0.62            | 0.271             | 0.707          | 4.25            | 0.363 |
| CASCADA           | 0.568          | 0.44            | 0.191             | 0.198          | 4.49            | 0.576 |

| AG News           | ROC AUC drop ↑ | Accuracy drop ↑ | Probability drop ↑ | Normalised WER ↓ | Log perplexity ↓ | NAD ↑ |
|-------------------|----------------|-----------------|-------------------|----------------|-----------------|------|
| Random search     | 0.474          | 0.83            | 0.680             | 0.445          | 5.21            | 0.355 |
| HotFlip           | 0.336          | 0.85            | 0.621             | 0.138          | 6.76            | 0.873 |
| MCMC              | 0.441          | 0.79            | 0.652             | 0.451          | 5.19            | 0.339 |
| CASCADA           | 0.447          | 0.87            | 0.705             | 0.248          | 6.33            | 0.685 |

| Healthcare Insurance | ROC AUC drop ↑ | Accuracy drop ↑ | Probability drop ↑ | Normalised WER ↓ | Log perplexity ↓ | NAD ↑ |
|----------------------|----------------|-----------------|-------------------|----------------|-----------------|------|
| Random search        | 0.828          | 0.71            | 0.248             | 0.390          | 4.90            | 0.377 |
| HotFlip              | 0.495          | 0.96            | 0.713             | 0.250          | 6.75            | 0.415 |
| MCMC                 | 0.354          | 0.61            | 0.195             | 0.394          | 4.81            | 0.330 |
| CASCADA              | 0.101          | 0.58            | 0.171             | 0.280          | 4.69            | 0.265 |
4.3 Quality of base models

We present model qualities for different classification and seq2seq models to motivate the selection of the model described in Section 3.1. We also compare various options for the training of the Deep Levenshtein distance defined in Section 3.2.3.

4.3.1 Classifier model quality

For the logistic regression with the TF-IDF features as inputs that we attack, the ROC scores for Transactions-AGE, Transcations-GENDER, Healthcare Insurance and AG News are 0.70, 0.74, 0.88, and 0.96 respectively. For multiclass problems we report the macro-average one-vs-rest ROC AUC.

4.3.2 Seq2seq model quality

The quality metrics for various datasets are presented in Table 4. We compare the basic sequence-to-sequence model (CopyNet without attention) to the sequence-to-sequence model with attention mechanism (CopyNet with attention) and the sequence-to-sequence model with the attention mechanism and masked training (Masked CopyNet with attention). Masked CopyNet with attention provides the best performance, so it has been selected as the seq2seq model for the generation of adversarial attacks on the logistic regression TF-IDF model.

Another model we need is Deep Levenshtein for the differentiable evaluation of the Levenshtein distance. The quality metrics for various datasets are presented in Table 5. We compare a variation from the initial paper (Basic) to the approaches with attention that imply embeddings learned using a seq2seq model (CopyNet with an attention encoder) with and without masking (Masked CopyNet with an attention encoder). Our results suggest that whilst the usage of embeddings pretrained from different problems can harm the performance of the model, the attention mechanism keeps the model performance at a level similar to that of the basic model.

| Model        | Transactions | Gender | Medical | AG News |
|--------------|--------------|--------|---------|---------|
| CopyNet      | 0.13         | 0.34   | 0.39    | 0.04    |
| +ATT         | 0.85         | 0.80   | 0.82    | 0.64    |
| +ATT+MASK    | 0.86         | 0.82   | 0.83    | 0.65    |

Table 4: CopyNet BLEU scores for different datasets, attention (ATT) and attention and masked (ATT+MASK) training. Masked CopyNet with attention produces the best BLEU score.

| Model        | Transactions | Gender | Medical | AG News |
|--------------|--------------|--------|---------|---------|
| BASE         | 0.16         | 0.16   | 0.20    | 0.21    |
| CN-ATT       | 0.14         | 0.17   | 0.19    | 0.20    |
| CN-ATT-MASK  | 0.14         | 0.15   | 0.18    | 0.20    |

Table 5: L1 distance scores for different datasets for various trainings of Deep Levenshtein approximation of the true Levenshtein distance: basic option (BASE), CopyNet with attention (CN-ATT) and CopyNet with attention and masking (CN-ATT-MASK) as encoders. Even with freezed encoders, CN-ATT(MASK) models perform on a level similar to basic option.

4.4 Main experiment for adversarial attacks

We run experiments to keep WER similar for the four considered approaches: random walk attack, MCMC walk attack, HotFlip, and CASCADA. We select hyperparameters to approximately match mean the WER scores for different approaches. We generate 100 sequences for each of the four approaches (so the comparison is fair) and select the best one according to the criterion described above.

In Table 6 we present results for proposed approaches whilst attacking an independent logistic regression model with TF-IDF features. In Table 7 we present results for proposed approaches whilst attacking an independent LSTM model with the end to end training of the embeddings. For adversarial attacks we use a seq2seq end to end model based on the LSTM architecture.

4.5 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 6; all types of attacks have comparable performances for our CASCADA approach.

4.6 Reliability study

In this section we examine how the selection of hyperparameters affects the performance of an adversarial attack. In particular, we run seq2seq models and observe the dependence of WER and the drop in accuracy for adversarial examples on the selection of hyperparameters. We have tried 599 different configurations in total for training seq2seq 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 the mean values of WER and the accuracy drop are inversely related for all considered hyperparameters: there is no silver bullet that provides an adversarial sequence for the initial sequence with corrupted probabilities for the true class, which is also similar to the initial sequence.

On the other hand, the varying of configurations has little effect on the quality of adversarial attacks. Thus, we conclude that the model is rather robust, and any appropriately chosen set of hyperparameters leads to a reasonable result.

5 Conclusion

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

We go in both directions and propose two approaches. The first approach is based on applying MCMC to generated sequences and the second approach uses surrogates for constructing gradient attacks. It turns out, that for considered applications that include NLP, bank card transactions, and healthcare our approaches show a reasonable performance based on values of common metrics for adversarial attacks and sequence distances and a new metric we propose that tries to examine if the generated sequence is both close and adversarial. Moreover, for our approaches we can limit the space of possible modifications, e.g. use only additions operations during an adversarial sequence generation.

At the core of our approaches lies a modern seq2seq architecture, which demonstrates an adequate performance for the tasks of interest. To obtain better results we adopt recent ideas from the NLP world, including masked training and the attention mechanism. The same ideas are useful whilst training surrogate models.

Table 6: Constrained adversarial attacks on Logistic regression with TF-IDF using various masking tokens for 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 |

References

[1] T. J. Sejnowski, *The deep learning revolution*. MIT Press, 2018.
[2] O. Sudakov, E. Burnaev, and D. Koroteev, “Driving digital rock towards machine learning: Predicting permeability with gradient boosting and deep neural networks,” *Computers & geosciences*, vol. 127, pp. 91–98, 2019.
[3] 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.
[4] 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.
[5] V. Khrulkov and I. Oseledets, “Art of singular vectors and universal adversarial perturbations,” in *IEEE CVPR*, pp. 8562–8570, 2018.
[6] W. E. Zhang, Q. Z. Sheng, A. Alhazmi, and C. Li, “Adversarial attacks on deep learning models in natural language processing: A survey,” *arXiv preprint arXiv:1901.06796*, 2019.
[7] W. Wang, B. Tang, R. Wang, L. Wang, and A. Ye, “A survey on adversarial attacks and defenses in text,” *arXiv preprint arXiv:1902.07285*, 2019.
[8] L. Sun, J. Wang, P. S. Yu, and B. Li, “Adversarial attack and defense on graph data: A survey,” *arXiv preprint arXiv:1812.10528*, 2018.
[9] A. Kurakin, I. J. Goodfellow, and S. Bengio, “Adversarial machine learning at scale,” *https://arxiv.org/abs/1611.01236*, 2017.
[10] S. Samanta and S. Mehta, “Towards crafting text adversarial samples,” *arXiv preprint arXiv:1707.02812*, 2017.
[11] B. Liang, H. Li, M. Su, P. Bian, X. Li, and W. Shi, “Deep text classification can be fooled,” *arXiv preprint arXiv:1704.08006*, 2017.
[12] J. Ebrahimi, A. Rao, D. Lowd, and D. Dou, “Hotflip: White-box adversarial examples for text classification,” in *56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 31–36, 2018.
Figure 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.

[13] M. Sato, J. Suzuki, H. Shindo, and Y. Matsumoto, “Interpretable adversarial perturbation in input embedding space for text,” in IJCAI, 2018.

[14] S. Moon, L. Neves, and V. Carvalho, “Multimodal named entity recognition for short social media posts,” in Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 852–860, 2018.

[15] S. Bowman, L. Vilnis, O. Vinyals, A. Dai, R. Jozefowicz, and S. Bengio, “Generating sentences from a continuous space,” in 20th SIGNLL Conference on Computational Natural Language Learning, pp. 10–21, 2016.

[16] C. Szegedy, W. Zaremba, I. Sutskever, J. B. Estrach, D. Erhan, I. Goodfellow, and R. Fergus, “Intriguing properties of neural networks,” in ICLR, 2014.

[17] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” arXiv preprint arXiv:1412.6572, 2014.

[18] D. Zügner, A. Akbarnejad, and S. Günnemann, “Adversarial attacks on neural networks for graph data,” in 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 2847–2856, 2018.

[19] N. Papernot, P. McDaniel, A. Swami, and R. Harang, “Crafting adversarial input sequences for recurrent neural networks,” in MILCOM 2016-2016 IEEE Military Communications Conference, pp. 49–54, IEEE, 2016.

[20] 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, 2016.

[21] 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.

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

[23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in NeurIPS, pp. 5998–6008, 2017.

[24] S. Wiseman and A. M. Rush, “Sequence-to-sequence learning as beam-search optimization,” in Empirical Methods in Natural Language Processing, pp. 1296–1306, 2016.

[25] W. Hastings, “Monte carlo sampling methods using markov chains and their applications,” Biometrika, pp. 97–109, 1970.

[26] R. Zhang, C. Li, J. Zhang, C. Chen, and A. G. Wilson, “Cyclical stochastic gradient mcmc for bayesian deep learning,” arXiv preprint arXiv:1902.03932, 2019.

[27] M. Cheng, J. Yi, H. Zhang, P.-Y. Chen, and C.-J. Hsieh, “Seq2Sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples,” arXiv preprint arXiv:1803.01128, 2018.

[28] Z. Zhukov, E. Golikov, and M. Kretov, “Differentiable lower bound for expected bleu score,” arXiv preprint arXiv:1712.04708, 2017.

[29] E. Ofitserov, V. Tsvetkov, and V. Nazarov, “Soft edit distance for differentiable comparison of symbolic sequences,” arXiv preprint arXiv:1904.12562, 2019.

[30] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” in ICLR, 2015.

[31] P. Li, W. Lam, L. Bing, and Z. Wang, “Deep recurrent generative decoder for abstractive text summarization,” in Conference on Empirical Methods in Natural Language Processing, pp. 2091–2100, 2017.

[32] M. Cheng, J. Yi, H. Zhang, P.-Y. Chen, and C.-J. Hsieh, “Seq2Sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples,” arXiv preprint arXiv:1803.01128, 2018.

[33] V. Zhukov, E. Golikov, and M. Kretov, “Differentiable lower bound for expected bleu score,” arXiv preprint arXiv:1712.04708, 2017.

[34] E. Ofitserov, V. Tsvetkov, and V. Nazarov, “Soft edit distance for differentiable comparison of symbolic sequences,” arXiv preprint arXiv:1904.12562, 2019.

[35] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” in ICLR, 2015.

[36] P. Li, W. Lam, L. Bing, and Z. Wang, “Deep recurrent generative decoder for abstractive text summarization,” in Conference on Empirical Methods in Natural Language Processing, pp. 2091–2100, 2017.

[37] D. Adiwardana, M.-T. Luong, D. R. So, J. Hall, N. Fiedel, R. Thoppilan, Z. Yang, A. Kulshreshtha, G. Nemade, Y. Lu, et al., “Towards a human-like open-domain chatbot,” arXiv preprint arXiv:2001.09977, 2020.

[38] R. Hu, J. Andreas, M. Rohrbach, T. Darrell, and K. Saenko, “Learning to reason: End-to-end module networks for visual question answering,” in IEEE ICCV, pp. 804–813, 2017.

[39] 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.

[40] J. Gu, Z. Lu, H. Li, and V. O. Li, “Incorporating copying mechanism in sequence-to-sequence learning,” in 54th Annual Meeting of the Association for Computational
[36] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “BLEU: a method for automatic evaluation of machine translation,” in *40th annual meeting on association for computational linguistics*, pp. 311–318. Association for Computational Linguistics, 2002.

[37] F. A. Gers, J. Schmidhuber, and F. Cummins, “Learning to forget: Continual prediction with lstm,” 1999.

[38] A. Graves, “Sequence transduction with recurrent neural networks,” *arXiv preprint arXiv:1211.3711*, 2012.

[39] X. Zhang, J. Zhao, and Y. LeCun, “Character-level convolutional networks for text classification,” in *NeurIPS*, pp. 649–657, 2015.

[40] “Sberbank Age transaction dataset.” [https://onti.ai-academy.ru/competition](https://onti.ai-academy.ru/competition) Accessed: 2020-07-02.

[41] “Bank gender transaction dataset.” [https://www.kaggle.com/c/python-and-analyze-data-final-project/data](https://www.kaggle.com/c/python-and-analyze-data-final-project/data) Accessed: 2020-07-02.