Sequence to Sequence Model for Anomaly Detection in Financial Transactions

Allan Inocêncio de Souza Costa ALLAN@CLOUDWALK.IO
Luis Silva LUIS@CLOUDWALK.IO
CloudWalk, Inc., 440 North Wolfe Road, Sunnyvale, CA, 94085 USA

Abstract

We propose the use of a character-level sequence-to-sequence based model to perform anomaly detection in financial transactions defined by $(request, response)$ pairs. We found empirically that we can use a single end-to-end model to learn patterns from a great variety of industry protocols with enough speed to perform real-time anomaly detection.

1. Introduction

At the heart of any payment business lies the concept of transaction. In this paper, a transaction means information exchange between two or more parties.

One typical work-flow is credit card payment using a point-of-sale device: we input all information of a sale in the machine, the device sends this information to an issuer backend which then replies with an approved/denied message.

In this context, it is valuable for the issuers to have the ability to detect high level patterns in transactions and also to identify unusual transactions. These anomalous transactions can contain hints about changes in application usage, man-in-the-middle attacks or even runtime errors on the back-end, as in this last case the response from the back-end will probably change.

In practice, we process simultaneously a great variety of transactions from different businesses using various protocols, so any manually coded pattern matching algorithm becomes infeasible. A better strategy is to automatically learn patterns from data using Machine Learning techniques.

Proceedings of the 33^{rd} International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48.

2. Model architecture

A transaction is a pair $(\mathbf{s}^{req}, \mathbf{s}^{res})$ of request $\mathbf{s}^{req} = (s_1^{req}, s_2^{req}, \ldots, s_{T_{req}}^{req})$ and response $\mathbf{s}^{res} = (s_1^{res}, s_2^{res}, \ldots, s_{T_{res}}^{res})$ sequences, where each sequence member is an integer token representing a symbol (a character in our case). We assume all request sequences have length T_{req} and all response sequences length T_{res}[1].

To learn transactions probabilities, we use a sequence-to-sequence model (Sutskever et al., 2014) where the task is to predict the response characters from the request ones. This is an encoder-decoder architecture, where we first encode the request as the output state of a Recurrent Neural Network (RNN) with Long-Short Term Memory (LSTM) cells (Hochreiter & Schmidhuber, 1997). We then use another RNN with LSTM cells to predict the response characters from the encoded request vector.

In Figure 1 we have a high level diagram of a sequence-to-sequence model. Notice that we have one LSTM network for the encoder and another for the decoder. Linking the two LSTM networks there is a vector $\mathbf{v}$ summarizing the request. The model is trained end-to-end to minimize the discrepancy between the decoder outputs (predicted values) and the next decoder input (true value).

An important aspect of our model is that the encoder vector $\mathbf{v}$ is presented to the decoder at each step. We found this mechanism introduces a huge speedup in training. It's not how the original sequence-to-sequence model of Sutskever worked, but it can be seen as a kind of *attention mechanism* in the spirit of (Bahdanau et al., 2014), although our mechanism is much simpler, due to the simpler nature of our problem.

[1] We can pad and trim sequences with different lengths to ensure they all have the same length. In this case, it is a good idea to prepend pad symbols to requests and to append pad symbols to responses, ensuring that their main content is not separated by any pad symbols.

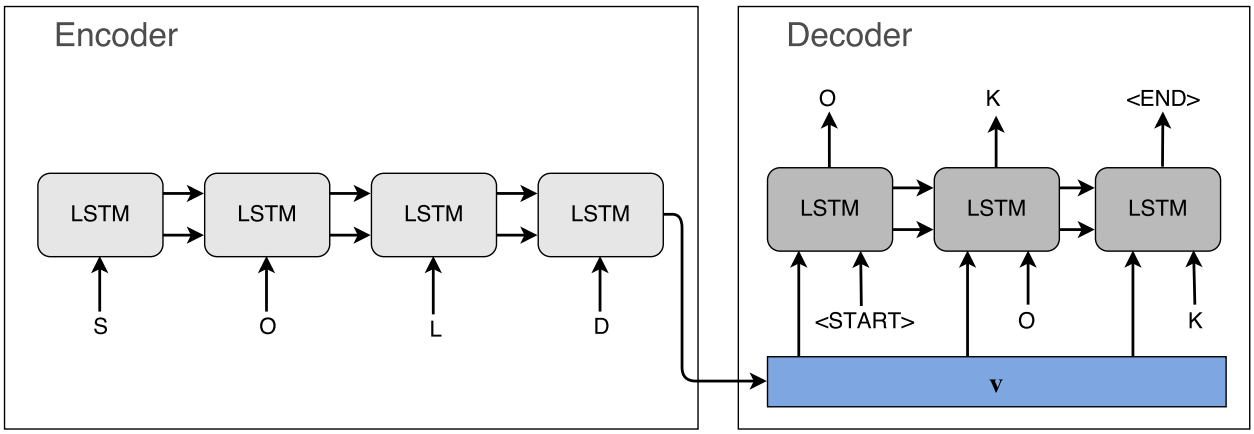

Figure 1. A high level sequence-to-sequence model. In this example, we are processsing a hypothetical transaction with request $\mathbf{s}^{req} = (S, O, L, D)$ and response $\mathbf{s}^{res} = (O, K)$. In practice we encode requests and responses as integer numbers and the sequences are much longer (300 elements in average).

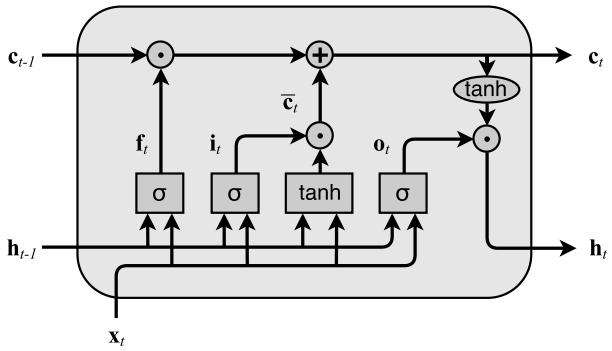

Figure 2. A LSTM cell at step t. Notice that it receives as inputs $\mathbf{c}_{t-1}$, $\mathbf{h}_{t-1}$ and $\mathbf{x}_t$ and outputs $\mathbf{c}_t$ and $\mathbf{h}_t$. The rectangles represent parametrized linear operations on their inputs followed by the specified activation function. Diagram created by the author inspired by the work of Chris Olah at http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Another important aspect of our work is that we don't really need the model to learn an exact model of all the transactions it sees. Instead, we hope it will learn about the general structure of the transactions, like its delimiters, fixed fields and average length, as this kind of template matching can deal with a great amount of anomaly detection.

In the next two subsections we are going to provide detailed information about our model.

2.1. Computing a transaction probability

To encode a request as a fixed size vector, we first initialize h_0 to the zero vector and iterate the following equations from $t = 1$ to $t = T_{req}$:

$$\mathbf{x}_t = \mathbf{M}\,[s_t^{req}] \tag{1}$$
$$\mathbf{i}_t = \sigma(\mathbf{W}^e_{ix}\mathbf{x}_t + \mathbf{W}^e_{ih}\mathbf{h}_{t-1} + \mathbf{b}^e_i) \tag{2}$$
$$\mathbf{f}_t = \sigma(\mathbf{W}^e_{fx}\mathbf{x}_t + \mathbf{W}^e_{fh}\mathbf{h}_{t-1} + \mathbf{b}^e_f) \tag{3}$$
$$\bar{\mathbf{c}_t} = \tanh(\mathbf{W}^e_{cx}\mathbf{x}_t + \mathbf{W}^e_{ch}\mathbf{h}_{t-1} + \mathbf{b}^e_c) \tag{4}$$
$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \bar{\mathbf{c}_t} \tag{5}$$
$$\mathbf{o}_t = \sigma(\mathbf{W}^e_{ox}\mathbf{x}_t + \mathbf{W}^e_{oh}\mathbf{h}_{t-1} + \mathbf{b}^e_o) \tag{6}$$
$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \tag{7}$$

In Equation (1) we retrieve the input's character embedding vector as the s_t^{req}-th row of embedding matrix $\mathbf{M}$. Equations (2) to (7) are the LSTM equations and we can see those operations summarized in Figure 2.

The encoded vector $\mathbf{v}$ is computed as a summary of the sequence of LSTM states:

$$\mathbf{v} = f(\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_T) \tag{8}$$

We experimented with two different functions f: the average function $f(\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_T) = \sum_{t=1}^{T} \mathbf{h}_t/T$ and the last state function $f(\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_T) = \mathbf{h}_T$. As we did not notice significant differences in practice, we chose to use the last state function in subsequent experiments.

With the request encoded as $\mathbf{v}$, we use another LSTM to decode this $\mathbf{v}$ generating, token by token, a response sequence. We do that by iterating the following equa-

tions from $t = 1$ to $t = T_{res}$:

$$\mathbf{x}_t = \mathbf{M}\left[s^{res}_{t-1}\right] \tag{9}$$

$$\mathbf{i}_t = \sigma(\mathbf{W}^d_{ix}\mathbf{x}_t + \mathbf{W}^d_{ih}\mathbf{h}_{t-1} + \mathbf{W}_{iv}\mathbf{v} + \mathbf{b}^d_i) \tag{10}$$

$$\mathbf{f}_t = \sigma(\mathbf{W}^d_{fx}\mathbf{x}_t + \mathbf{W}^d_{fh}\mathbf{h}_{t-1} + \mathbf{W}_{fv}\mathbf{v} + \mathbf{b}^d_f) \tag{11}$$

$$\bar{\mathbf{c}}_t = \tanh(\mathbf{W}^d_{cx}\mathbf{x}_t + \mathbf{W}^d_{ch}\mathbf{h}_{t-1} + \mathbf{W}_{cv}\mathbf{v} + \mathbf{b}^d_c) \tag{12}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \bar{\mathbf{c}}_t \tag{13}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}^d_{ox}\mathbf{x}_t + \mathbf{W}^d_{oh}\mathbf{h}_{t-1} + \mathbf{W}_{ov}\mathbf{v} + \mathbf{b}^d_o) \tag{14}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \tag{15}$$

$$\mathbf{y}_t = \mathbf{W}\mathbf{h}_t + \mathbf{b} \tag{16}$$

$$\mathbf{z}_t = \text{softmax}(\mathbf{y}_t) \tag{17}$$

There are some differences between the decoder equations above and the encoder ones. First, at each step the LSTM receives the encoder vector $\mathbf{v}$ as an extra input. The second difference is the use of a softmax function to compute a probability vector $\mathbf{z}_t$ such that $\mathbf{z}_t\left[s^{res}_t\right]$ is the probability that the next character will be s^{res}_t:

$$\mathbf{z}_t\left[s^{res}_t\right] = p(r^{res}_t | s^{res}_1, \ldots, s^{res}_{t-1}, \mathbf{s}^{req}). \tag{18}$$

The conditionals in $p(s^{res}_t | s^{res}_1, \ldots, s^{res}_{t-1}, \mathbf{s}^{req})$ are included because $\mathbf{z}_t$ computation is dependent on the request (through $\mathbf{v}$) and on the previous response tokens though $\mathbf{h}_{t-1}$. The softmax function is a function from vectors to vectors with some nice properties that allow us to interpret its output as a probability vector, including its components summing to 1 and the boundedness of each component to the interval $[0, 1]$. The function is defined by:

$$\text{softmax}(\mathbf{z}) = \frac{e^{\mathbf{z}}}{\sum_i e^{z_i}} \tag{19}$$

From the probabilities computed at each step, we are able to compute the whole transaction probability according to the conditional probability law:

$$p(\mathbf{s}^{res} | \mathbf{s}^{req}) = \prod_{t=1}^{T_{res}} p(s^{res}_t | s^{res}_1, \ldots, s^{res}_{t-1}, \mathbf{s}^{req}) = \tag{20}$$

$$= \prod_{t=1}^{T_{res}} \mathbf{z}_t\left[s^{res}_t\right] \tag{21}$$

2.2. Training a model to find the best parameters

Assume we have a vocabulary[2] of size V, an embedding dimension E and a LSTM cell dimension H.

[2]We are keeping the traditional naming here, but we remind the reader that our vocabulary is in fact made of distinct characters.

Table 1. All model's parameters separated by encoder and decoder. At the bottom we show the total amount of parameters, which is simply the sum of all dimensions.

Encoder		Decoder	
Symbol	Dimension	Symbol	Dimension
$\mathbf{M}$	$V \times E$	$\mathbf{M}$	$V \times E$
$\mathbf{W}^e_{ix}$	$H \times E$	$\mathbf{W}^d_{ix}$	$H \times E$
$\mathbf{W}^e_{ih}$	$H \times H$	$\mathbf{W}^d_{ih}$	$H \times H$
$\mathbf{b}^e_i$	H	$\mathbf{b}^d_i$	H
$\mathbf{W}^e_{fx}$	$H \times E$	$\mathbf{W}^d_{fx}$	$H \times E$
$\mathbf{W}^e_{fh}$	$H \times H$	$\mathbf{W}^d_{fh}$	$H \times H$
$\mathbf{b}^e_f$	H	$\mathbf{b}^d_f$	H
$\mathbf{W}^e_{cx}$	$H \times E$	$\mathbf{W}^d_{cx}$	$H \times E$
$\mathbf{W}^e_{ch}$	$H \times H$	$\mathbf{W}^d_{ch}$	$H \times H$
$\mathbf{b}^e_c$	H	$\mathbf{b}^d_c$	H
$\mathbf{W}^e_{ox}$	$H \times E$	$\mathbf{W}^d_{ox}$	$H \times E$
$\mathbf{W}^e_{oh}$	$H \times H$	$\mathbf{W}^d_{oh}$	$H \times H$
$\mathbf{b}^e_o$	H	$\mathbf{b}^d_o$	H
-	-	$\mathbf{W}^d_{iv}$	$H \times H$
-	-	$\mathbf{W}^d_{cv}$	$H \times H$
-	-	$\mathbf{W}^d_{cv}$	$H \times H$
-	-	$\mathbf{W}^d_{ov}$	$H \times H$
-	-	$\mathbf{W}$	$H \times V$
-	-	$\mathbf{b}$	V
Total	$8(H \times E + H) + 12(H \times H)$ $+V \times E + H \times V + V$		

In Table 1 we collect all parameters in our model with their respective dimensions and a brief description.

We obtain the parameters through training, in which we feed the model transactions data and give it the task of correctly predicting at each decoder step the correct character in the response at the given position. From these predictions, computed using equation (18), we compute a cross-entropy loss for the entire transaction:

$$L(\mathbf{s}^{req}, \mathbf{s}^{res}) = -\sum_{t=1}^{T_{res}} \frac{\log(\mathbf{z}_t\left[s^{res}_t\right])}{T_{res}}. \tag{22}$$

The parameters' gradients are computed in batches from this loss function using backpropagation (Rumelhart et al., 1988) and are updated accordingly by the Adam optimization algorithm (Kingma & Ba, 2014).

For the experiments in this paper we have used Chainer (Tokui et al., 2015) to implement our model. The parameters gradients are automatically computed from the computational graph.

2.3. Computing anomaly scores from probabilities

As our system should be able to work with partial information in transactions from multiple probability distributions, we cannot simply perform a threshold on a transaction's probability to flag it as anomalous or not. That occurs because each distribution has a different information pattern in the request that can be used to predict the response, leading to different mean probabilities for distinct distributions, so that a transaction with 50% probability might not be anomalous for a given distribution (for example, if we see few transactions from this distribution in our training dataset).

To be able to perform anomaly detection using a single model in such a diversified environment, where even a single distribution among many can be multimodal, we devised a technique based on an average probability computed by using similar transactions only.

To find similar transactions, we use the encoded vectors $\mathbf{v}$. We assume that similar transactions have close encodings, so given two encoded transactions as vectors $\mathbf{v}_1$ and $\mathbf{v}_2$, we can compute a similarity score based on the Euclidean distance between them:

$$\text{similarity}(\mathbf{v}_1, \mathbf{v}_2) = \frac{1}{1 + (\mathbf{v}_1 - \mathbf{v}_2)^T(\mathbf{v}_1 - \mathbf{v}_2)} \quad (23)$$

As the encoded vectors are bounded, we have $0 < (\mathbf{v}_1 - \mathbf{v}_2)^T(\mathbf{v}_1 - \mathbf{v}_2) < \infty$, so that $0 < \text{similarity}(\mathbf{v}_1, \mathbf{v}_2) \leq 1$, where the similarity is closer to zero the greater the distance between the transactions and is closer to one the lesser the distance between them.

The average probability of a given transaction with encoded vector $\mathbf{v}$ is an average weighted by the similarity scores:

$$\bar{p}(\mathbf{s}^{req}, \mathbf{s}^{res}) = \frac{\sum_k \text{similarity}(\mathbf{v}, \mathbf{v}_k) p(\mathbf{s}_k^{req} | \mathbf{s}_k^{res})}{\sum_k \text{similarity}(\mathbf{v}, \mathbf{v}_k)}. \quad (24)$$

The range of the summations can be the entire dataset, all transactions in the same distribution as the given transaction (inferred from some metadata) or even the top n most similar transactions. For this work we chose the second option.

We can then compute an anomaly score based on this average probability:

$$\text{score}(\mathbf{s}^{req}, \mathbf{s}^{res}) = \max\left(1 - \frac{p(\mathbf{s}^{req} | \mathbf{s}^{res})}{\bar{p}(\mathbf{s}^{req} | \mathbf{s}^{res})}, 0\right). \quad (25)$$

The anomaly score interpretation is that if a transaction has, for instance, an anomaly score of 80%, then its probability as computed by the model is only 20% of the average probability for this transaction. So the higher the score, the more deviant the transaction is from the average and we can threshold this anomaly score to decide if a given transaction is anomalous.

In this work we flagged as anomalous transactions having $\text{score}(\mathbf{s}^{req}, \mathbf{s}^{res}) > 0.5$.

3. Cerebro

Based on this paper's ideas, we developed a product called Cerebro, which we describe with detail in this section.

3.1. Preprocessing

For each transaction, we extract its request and response messages along with an extra field which is an identifier for the company that processed the transaction, which we call its acronym.

We preprocess the dataset so that each request has 300 characters and each response has 350 characters. We trim transactions with more characters and pad the ones with less. After that, we build mappings from characters to integer identifiers, storing them in volatile memory, but also saving them with the trained parameters, as they are crucial for online anomaly detection.

3.2. Training

The production version of Cerebro is trained once a day and uses the latest 150000 transactions processed by our company as its training dataset, of which 5% is kept for validation. We use a vocabulary size $V = 100$, embedding dimension $E = 128$ and LSTM cells of size $H = 192$. The relatively small number of parameters allows the model to be trained in around one hour using a laptop with a GPU Nvidia GTX 965M and an Intel Core i7-4720HQ CPU with 2.60 GHz $\times$ 8 cores.

We use Dropout (Srivastava et al., 2014) layers with early training stop to avoid overfitting.

We compute forward and backward passes for mini-batches of transactions, usually containing between 128 and 256 transactions. Before each epoch pass (a full pass in the dataset) we randomize our dataset.

An important point to notice is that we perform validation only on the LSTM training, not on the anomaly scores, as we do not have labeled data, making it difficult to provide simple objective metrics for the model's effectiveness. Nonetheless, we are running Cerebro with production data for some time and we have seen qualitatively that the anomalies found are meaningful.

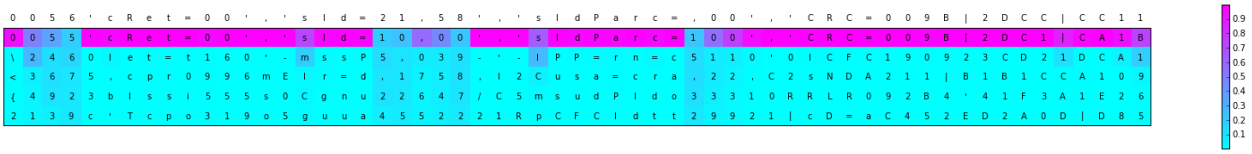

Figure 3. An example of predictions from our model. In the top we have the real expected response. The following lines are the top five predictions after each response's character is presented to the model. The colors represents the probabilities as per the legend.

3.3. Results

We have some real examples in Table 2 (with only some obfuscation to preserve anonymity). In the first and second group of transactions the anomalies are just rare transactions from totally distinct applications that occurred when the merchant or the customer entered a wrong password. The last group is more interesting, as our model caught a subtle change in the typical request ending: the usual behavior is to have in the last field something like `ps=4:1;`, but then for some reason we got a composite value `ps=4:2;5:1`, triggering an anomaly. We would like also to point out how different these groups of transactions are and yet they are just a small sample of the diversity our single model sees in a production environment.

Another way to visualize the model's inner workings is by inspecting the top five predictions of the model after each response's character is presented to it. We have an example in Figure 3, were we can see that the model learned with confidence to predict the fixed keys in a response pattern it saw many times. However, fields with more diverse outputs, like the value of the transaction, have much less confidence, as expected.

Yet another way to understand the model is by analyzing its computed requests' activations. We can use t-Distributed Stochastic Neighbor Embedding (t-SNE) (van der Maaten & Hinton, 2008) to project this high-dimensional data to two dimensions. We can see in Figure 4 that the model learns to cluster request data. Each circle represent a transaction, where each color corresponds to a different company and the circle size is proportional to the transaction probability as computed by our sequence-to-sequence model. Here we have the stronger argument in favor of our anomaly score computation as described in section 2.3, as we can see clearly from the figure that transactions with similar probabilities are clustered together. Further inspection also shows that transactions with similar contents are clustered together.

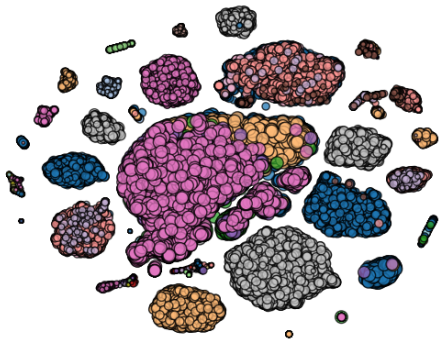

Figure 4. Projected activations using TSNE. Different colors represent transactions from different companies and circles sizes are proportional to transactions probabilities.

4. Conclusion

We have empirical results that show it is feasible to use character based sequence-to-sequence models to perform anomaly detection in transactions.

The proposed model is trained unsupervised, meaning that there is no need to show examples of anomalous transactions to train the model. This is a nice property, as it allows the model's usage with little effort in terms of human resources. It is important to have data available for training, although our experiments show that with 50000 transactions it's already possible to train good models.

Cerebro has been running in production consistently for some months now. We are planning to release it to the public in the near future.

Acknowledgments

I would like to thank David Kelleher for the English review of my original manuscript.

References

Bahdanau, Dzmitry, Cho, Kyunghyun, and Bengio, Yoshua. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014. URL http://arxiv.org/abs/1409.0473.

Hochreiter, Sepp and Schmidhuber, Jürgen. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL http://dx.doi.org/10.1162/neco.1997.9.8.1735.

Kingma, Diederik P. and Ba, Jimmy. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL http://arxiv.org/abs/1412.6980.

Rumelhart, David E., Hinton, Geoffrey E., and Williams, Ronald J. Neurocomputing: Foundations of research. chapter Learning Representations by Back-propagating Errors, pp. 696–699. MIT Press, Cambridge, MA, USA, 1988. ISBN 0-262-01097-6. URL http://dl.acm.org/citation.cfm?id=65669.104451.

Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

Sutskever, Ilya, Vinyals, Oriol, and Le, Quoc V. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014. URL http://arxiv.org/abs/1409.3215.

Tokui, Seiya, Oono, Kenta, Hido, Shohei, and Clayton, Justin. Chainer: a next-generation open source framework for deep learning. In *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Twenty-ninth Annual Conference on Neural Information Processing Systems (NIPS)*, 2015. URL http://learningsys.org/papers/LearningSys_2015_paper_33.pdf.

van der Maaten, Laurens and Hinton, Geoffrey. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008. URL http://www.jmlr.org/papers/v9/vandermaaten08a.html.

Table 2. Examples of typical transactions followed by anomalous ones. We also show the respective probabilities, average probabilities and anomaly scores (in red if anomalous).

request	46068007,CLOUDWALK,Fechamento,D2******F1528E20,367,<sn>529-***-***</sn>
response	0062'cRet=00','msg=LOTE FECHADO COM SUCESSO!','CRC=FB31-34A7-2DC1'
p	0.562
$\bar{p}$	0.608
score	0.075
request	46068007,CLOUDWALK,Fechamento,D2******F1528E20,367,<sn>529-***-***</sn>
response	0060'cRet=99','msg=Senha lojista invalida!','CRC=FB31-34A7-2DC1'
p	0.268
$\bar{p}$	0.597
score	0.551
request	0062@CLOUD@5400054@32000082@4093@61@6300000202517978@1100@2,00
response	0632@5400070@OK@#DDTOTAL - 11/04/2016 21:29:48#COMPROVANTE DE VENDA# #AUTORIZACAO N. 5200002#MERCADINHO ******#AV ***** *****, 951#MANAUS - AM#VALOR: R$ 2,00#CARTAO: **** **** **** 7978# # AUTORIZADO COM SENHA # # # NAO E DOCUMENTO FISCAL # 1A VIA - ESTABELECIMENTO # # # # # -=CORTE=-#DDTOTAL - 11/04/2016 21:29:48#COMPROVANTE DE VENDA# #AUTORIZACAO N. 5200002#MERCADINHO ******#AV ***** *****, 951#MANAUS - AM# VALOR: R$ 2,00#CARTAO: **** **** **** 7978#SALDO DISPONIVEL:## SEGMENTO - TOTAL - DISP. #C.ALIMENTACA- R$0,0- R$193,75## # NAO E DOCUMENTO FISCAL # 2A VIA - CLIENTE # # # # #
p	0.586
$\bar{p}$	0.158
score	0.000
request	0062@CLOUD@5400077@32000730@5728@61@6300000102474452@5479@8,00
response	0034@5400023@ERR@SENHA NAO CONFERE
p	0.112
$\bar{p}$	0.713
score	0.696
request	POST /type9.php type=9\&ch=2402\&bb=3124\&acc=93\&pos=106\&ps=4:1;
response	000D;9000;200;200
p	0.174
$\bar{p}$	0.158
score	0.000
request	POST /type9.php type=9\&ch=2299\&bb=3650\&acc=93\&pos=107\&ps=4:2;5:1;
response	000E;9000;650;2350
p	0.038
$\bar{p}$	0.125
score	0.696