Predicting Shopping Behavior with Mixture of RNNs

Arthur Toth
Rakuten Institute of Technology
Boston, Massachusetts 02110
arthur.toth@rakuten.com

Louis Tan
Rakuten Institute of Technology
Boston, Massachusetts 02110
ts-louis.tan@rakuten.com

Giuseppe Di Fabbrizio
Rakuten Institute of Technology
Boston, Massachusetts 02110
difabbrizio@gmail.com

Ankur Datta
Rakuten Institute of Technology
Boston, Massachusetts 02110
ankur.datta@rakuten.com

ABSTRACT

We compare two machine learning approaches for early prediction of shoppers' behaviors, leveraging features from clickstream data generated during live shopping sessions. Our baseline is a mixture of Markov models to predict three outcomes: purchase, abandoned shopping cart, and browsing-only. We then experiment with a mixture of Recurrent Neural Networks. When sequences are truncated to 75% of their length, a relatively small feature set predicts purchase with an F-measure of 0.80 and browsing-only with an F-measure of 0.98. We also investigate an entropy-based decision procedure.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; • **Applied computing** → **Online shopping**;

KEYWORDS

mining/modeling search activity, click models, behavioral analysis

ACM Reference format:
Arthur Toth, Louis Tan, Giuseppe Di Fabbrizio, and Ankur Datta. 2017. Predicting Shopping Behavior with Mixture of RNNs. In *Proceedings of SIGIR 2017 eCom, Tokyo, Japan, August 2017,* 5 pages.
DOI:

1 INTRODUCTION

Recent e-commerce forecast analysis estimates that more than 1.77 billion users will shop online by the end of 2017 [11]. Although this is impressive growth, conversion rates for online shoppers are substantially lower than rates for traditional brick-and-mortar stores.

Consumers shopping on e-commerce web sites are influenced by numerous factors and may decide to stop the current session, leaving products in their shopping carts. Once a user has interacted with a shopping cart, abandonment rates range between 25% and 88%, significantly reducing merchants' selling opportunities [15]. Several potential *purchase inhibitors* have been analyzed in the online shopping literature [5, 8]. Main causes include concerns about costs, perceived decision difficulty, and selection conflicts due to a large number of similar choices.

Early shopping abandonment detection may allow mitigation of purchase inhibitors by enabling injection of new content into live web browsing sessions. For instance, it could trigger a discount offer or change the product search strategy to retrieve more diverse options and simplify the consumer's decision process.

This paper considers real web interactions from a US e-commerce subsidiary of Rakuten (楽天株式会社) to predict three possible outcomes: purchase, abandoned shopping cart, and browsing-only. To early detect outcomes, we consider clues left behind by shoppers that are encoded into *clickstream* data and logged during each session. Clickstream data is used to experiment with mixtures of high-order Markov Chain Models (MCMs) and mixtures of Recurrent Neural Networks (RNNs) that use the Long Short-Term Memory (LSTM) architecture. We compare and contrast each model on sequences truncated at different lengths and report on precision, recall, and F-measures. We also show F-measures from using an entropy-based decision procedure that is usable in a live system.

2 RELATED WORK

We treat predicting user behavior from clickstream data as sequence classification, which is broadly surveyed by Xing et al. [14], who divide it into feature-based, sequence distance-based, and model-based methods. Previous feature-based work on clickstream classification includes the random forest used by Awalker et al. [2], the deep belief networks and stacked denoising auto-encoders by Viera [12], and recurrent neural networks by Wu et al. [13]. Previous distance-based work includes the large margin nearest neighbor approach by Pai et al. [10]. Previous model-based work by Bertsimas et al. [4] used a mixture of Markov chains.

Our baseline approach is based on the latter work, whereas our new approach uses a mixture of RNNs. Although Wu et al. [13] used RNNs, their approach is not applicable to our scenario, since its bi-directional RNN uses entire clickstream sequences. Our goal is to classify incomplete sequences. Also their model is not a mixture.

Finally, our approach differs from most others in its use of a ternary classification scheme. We classify clickstreams as either being a *purchase*, *abandon* or *browsing-only* session instead of just *purchase* and *non-purchase*.

SIGIR 2017 eCom, Tokyo, Japan

DOI:

Table 1: Clickstream data distribution.

Clickstream outcome	Count	%	Average Length $\bar{m}_i$	Median Length
ABANDON	29,371	14.7%	8.2	5
BROWSING-ONLY	128,450	64.6%	16.6	6
PURCHASE	41,115	20.7%	8.4	5
Total	198,936	100%		

3 CLICKSTREAM DATA

We consider clickstream data collected over two weeks and consisting of $n_0 = 1,560,830$ sessions. A *session* $\mathbf{S}_i, i = 1, 2, \ldots, n_0$, is a chronological sequence of recorded *page views*, or "*clicks*." Let $V_{i,j}$ be the jth page view of session i so that $\mathbf{S}_i = (V_{i,1}, V_{i,2}, \ldots V_{i,m_i})$, and m_i is defined as the *length* of session i. We exclude session i from our experiments if $m_i < 4$, after which $n = 198,936$ sessions remain. Sessions with purchases are truncated before the first purchase confirmation. Table 1 summarizes the n sessions.

Our experiments use both the *page type* and *dwell time* of $V_{i,j}$. The page type of $V_{i,j}$, denoted as $P_{i,j}$, belongs to one of eight categories, including search pages, product view pages, login pages, etc. The dwell time, $D_{i,j}$, of $V_{i,j}$ is the amount of time the user spends viewing the page, and is not available until the $(j+1)$th page view. After the jth page view, the clickstream data gathered for session i is given by $\mathbf{S}_i|_j = ((P_{i,1}, D_{i,0}), (P_{i,2}, D_{i,1}), \ldots, (P_{i,j}, D_{i,j-1}))$ where $D_{i,0}$ is undefined, i.e., $D_{i,0} = \emptyset$. To reduce sparsity, dwell times were placed in 8 bins, evenly spaced by percentiles.

4 MODELING APPROACHES

Our goal is to classify customer behavior into final decision categories. In particular, clickstream sequences receive one of the following labels: PURCHASE, if the sequence leads to an item purchase; ABANDON, if an item was left in the shopping cart, but there was no purchase; and BROWSING-ONLY, when the shopping cart was not used. The final two categories can be combined to investigate PURCHASE vs. NON_PURCHASE behavior. In preliminary studies, the ABANDON sequences were much more similar to the PURCHASE sequences than to the BROWSING-ONLY sequences, so having three categories helped account for some of the confusability of the data. Our eventual goal of applying our models in a live system adds a constraint. The classifier must work for incomplete sequences, without using data from the "future".

4.1 Mixture of High-Order Markov Chains

Bertsimas et al. [4] modeled a similar problem with a mixture of Markov chains, using *maximum a posteriori* estimation to predict sequence labels. In this approach, the training data is partitioned by class and separate Markov chain models are trained for each one. The resultant models can estimate class likelihoods from sequences. Let C_i be a random variable representing the outcome of session $\mathbf{S}_i$, where $C_i \in \Omega$, and Ω is the set of possible clickstream outcomes. The models would be used to estimate the likelihoods $P(\mathbf{S}_i|C_i = \omega)$ for all $\omega \in \Omega$. Using Bayes' theorem and the law of total probability, the class posteriors for each of the three classes can be estimated by the equation

$$P(C_i = \omega|\mathbf{S}_i) = \frac{P(\mathbf{S}_i|C_i = \omega)P(C_i = \omega)}{\sum_{\omega \in \Omega} P(\mathbf{S}_i|C_i = \omega)P(C_i = \omega)}$$

with the prior, $P(C_i = \omega)$, estimated from counts.

This model fits the problem's constraints, because likelihoods can be produced from subsequences without using "future" clicks. Although each chain is trained only on click data, separating data by class implicitly conditions them on class.

Taking inspiration from the Automatic Speech Recognition (ASR) community and similarities to "Language Modeling", we adapted some of their more recent techniques to our problem. In preliminary experiments, 5-grams performed better than shorter chains, so we used them. Longer chains cause greater sparsity, so we addressed this with Kneser-Ney smoothing, which performed best in a study of language modeling techniques [6]. We used the MITLM toolkit to train the Markov chains [7].

4.2 Recurrent Neural Networks

Taking further inspiration from the ASR community, we replaced the Markov-chains in our mixtures with RNNs. Earlier language modeling work used feed-forward artificial neural networks [3], but RNNs have performed better recently, both in direct likelihood metrics and in overall ASR task metrics [9, 16]. Click-stream data differs from ASR text, and our mixture model differed from the typical ASR approach, so it was unclear whether RNNs would help in our scenario.

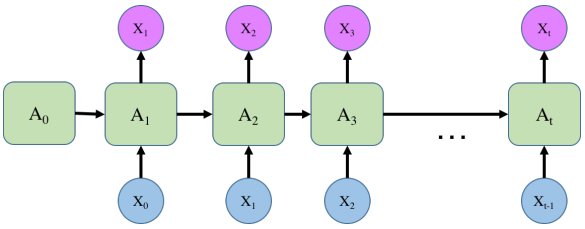

Figure 1: Simple RNN to Predict Following Token

Earlier RNN-based language models used the "simple recurrent neural network" architecture [9]. The underlying idea, depicted in Figure 1, is that an input sequence, represented by $\{X_0, ..., X_{t-1}\}$, is connected to a recurrent layer, represented by $\{A_0, ...A_t\}$, which is also connected to itself at the next point in time. The recurrent layer is also connected to an output layer. In our models, each RNN tries to predict the next input, X_{n+1}, after each input, X_n.

"Simple" RNN-based language models for ASR were outperformed by RNNs using the Long Short-Term Memory (LSTM) configuration and "drop-out" [16]. LSTM addressed the "vanishing gradient" and "exploding gradient" problems. "Drop-out" addressed overfitting by probabilistically ignoring the contributions of non-recurrent nodes during training.

In LSTM RNNs, some nodes function as "memory cells". Some connections retrieve information from them, and others cause them to forget. The LSTM equations are [16]:

$$LSTM: h_t^{l-1}, h_{t-1}^l, c_{t-1}^l \rightarrow h_t^l, c_t^l$$

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} sigm \\ sigm \\ sigm \\ sigm \end{pmatrix} T_{2n,4n} \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^{l} \end{pmatrix}$$

$$c_t^l = f \odot c_{t-1}^l + i \odot g$$

$$h_t^l = o \odot tanh(c_t^l)$$

where h and c represent hidden states and memory cells, subscripts refer to time, superscripts refer to layers, $T_{n,m}$ is an affine transform from R^n to R^m, $\odot$ refers to element-wise multiplication, and $sigm$ and $tanh$ are applied to each element.

Since LSTM RNNs with drop-out worked much better than Markov chains for ASR, we replaced the Markov chains with them in our clickstream mixture models. Additional work was necessary, since our scenario did not exactly match language modeling. During training, input tokens were still used to predict following tokens. During testing, however, our goal was the sequence probabilities. These were calculated from token probabilities present in intermediate softmax layers in each LSTM model. Due to the network architecture, "future" events were not used for this. We used TensorFlow to implement our LSTM RNNs [1].

5 EXPERIMENTS

We experimented on the dataset described in Section 3 and summarized in Table 1. It was partitioned into an 80% training/20% testing split, using *stratified sampling* to retain class proportions.

All RNNs were trained for 10 epochs, using batches of 20 sequences. We tested 16 combinations of the remaining parameters. The number of recurrent layers was 1 or 2, the keep probability was 1 or 0.5, and the hidden state size was 10, 20, 40, or 80. For a particular mixture model, all the RNNs used the same parameter values.

5.1 Results

For each model, we evaluated the prediction performance by truncating the page view sequences at different lengths. Table 2 shows the results for the mixture of Markov models, and for one of the mixture of RNN trials. Although we tested 16 different RNN parameter combinations, results were so similar that we are only reporting on one of them.

Table 2 reports precision, recall, and F1-measure for each specific sequence outcome when considering 25%, 50%, 75%, and 100% of the total length of the sequence. For instance, when splitting at 50%, the Markov chain model can predict a PURCHASE with a 0.42 precision and 0.11 recall, resulting in an overall F1-measure of 0.17. For the same conditions, the RNN-based model reaches a precision of 0.82 with 0.71 recall and an F1-measure of 0.76. We also report the accuracy when randomly selecting a class based on the prior distribution of the clickstream corpus. RNN mixture components substantially outperform Markov chain components. This is particularly evident from Figure 2, which shows the F1-measure by sequence length for the mixtures of MCMs (dotted line) and of RNNs (solid line). Both models monotonically increase performance as the model observes more data from 25% splits to 100%, but the mixture of RNNs has an immediate edge even with the short 25% sequences. The MCMs present similar F1-measures for the majority class (*i.e.*, BROWSING-ONLY), but it is penalized

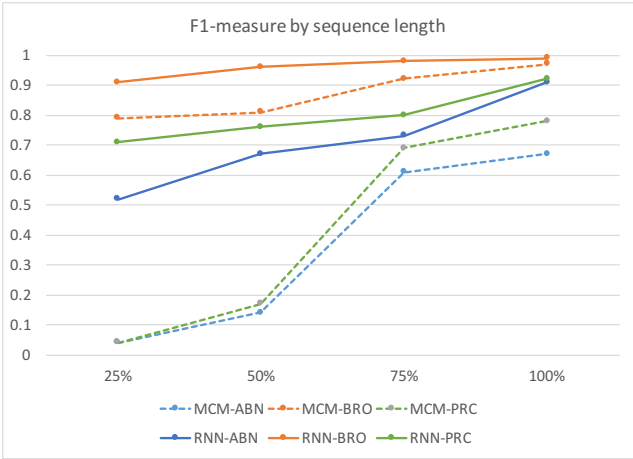

Figure 2: F1-measure by sequence length for mixtures of Markov Chain Models (MCM) and RNNs for each label: ABN=abandoned; BRO=browsing-only; PRC=purchase.

by the lack of data for the less represented sequences (*i.e.*, 14.7% for ABANDON and 20.7% for PURCHASE). RNNs instead generalize better due to the memory component that can model long-distance dependencies.

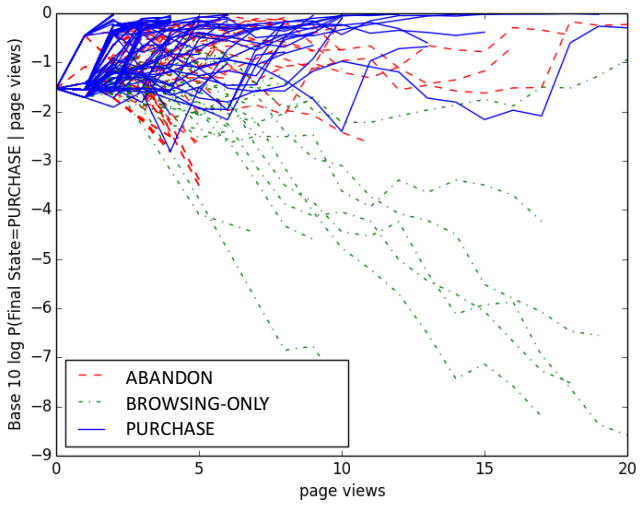

Figure 3: Log-probability trajectories from the MCM mixture, progressing along page view sequences for each class

Similarly to Bertsimas et al. [4], in Figure 3, we plot 100 log-probability trajectories, with lengths from 2 through 20 page views, estimated by the MCM mixture along page view sequences for each class. This plot demonstrates how probabilities evolve during interactions and how confident each model is compared to others.

The legend in Figure 3 corresponds to the true final state labels of the test examples: dashed red lines for ABANDON sequences, dashed and dotted green lines for BROWSING-ONLY sequences, and solid blue lines for PURCHASE sequences. Ideally, the model

Table 2: Precision (P), Recall (R), and F1-measure (F) for MCM and RNN mixtures by sequence length.

Mixture Type	Sequence Length	25%			50%			75%			100%			
MCM	Final State	P	R	F1	P	R	F1	P	R	F1	P	R	F1	Random
	ABANDON	0.44	0.02	0.04	0.45	0.09	0.14	0.72	0.52	0.61	0.70	0.64	0.67	0.21
	BROWSING-ONLY	0.66	0.99	0.79	0.69	0.98	0.81	0.89	0.97	0.92	0.99	0.95	0.97	0.64
	PURCHASE	0.37	0.02	0.04	0.42	0.11	0.17	0.72	0.67	0.69	0.71	0.86	0.78	0.15
RNN	Final State	P	R	F1	P	R	F1	P	R	F1	P	R	F1	Random
	ABANDON	0.75	0.39	0.52	0.73	0.62	0.67	0.74	0.72	0.73	1.00	0.84	0.91	0.21
	BROWSING-ONLY	0.84	0.99	0.91	0.92	0.99	0.96	0.97	0.99	0.98	1.00	0.98	0.99	0.64
	PURCHASE	0.80	0.64	0.71	0.82	0.71	0.76	0.82	0.78	0.80	0.86	1.00	0.92	0.15

would score the PURCHASE sequences (solid blue lines) high, and the other sequences low, and the earlier the distinction could be made, the better. Looking at this figure, there does appear to be some level of discrimination between the categories. In general, the BROWSING-ONLY sequences seem more separable from the PURCHASE sequences than the ABANDON sequences, as expected.

Although Table 2 can be used to compare other work [10], it depends on sequence lengths. A useful live system must predict final actions before sequences are complete and needs a decision process for accepting the predicted label. We experimented with taking the highest scoring label once the entropy fell below a threshold. Figure 4 shows the F1-measures at different thresholds, which are proportions of the maximum possible entropy. Since the entropy might not drop below the threshold, it is important to consider how many sequences have predicted labels. When we calculated F1-measures, sequences without predictions were counted as misses. Figure 5 shows the number of sequences where predictions were made based on entropy threshold. Again, the horizontal axis represents the proportion of the maximum possible entropy value. The vertical axis represents the number of sequences where a decision can be made based on threshold crossing. As an example, a threshold of 0.55 led to reasonable F1-measures while producing predictions for 99% of the sequences before they were complete. Choosing higher entropy thresholds allows decisions to be made for more sequences, but performance can suffer since decisions can be made while class probabilities are more uniform and confidence is lower. Choosing lower entropy thresholds forces the class probabilities to be more distinct, which leads to more confident decisions, but performance starts to suffer when fewer sequences receive decisions. In practice, the threshold would be tuned on held-out data.

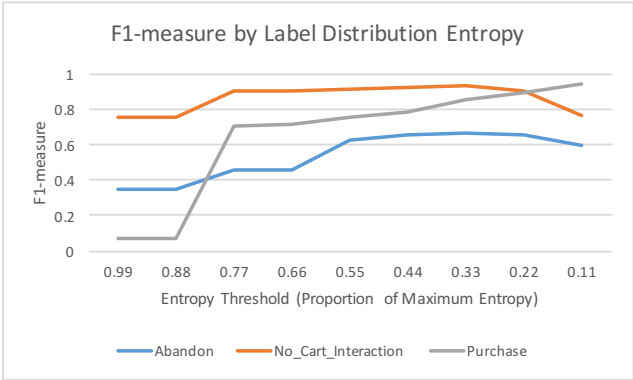

Figure 4: F1-measures at Entropy Thresholds for each class

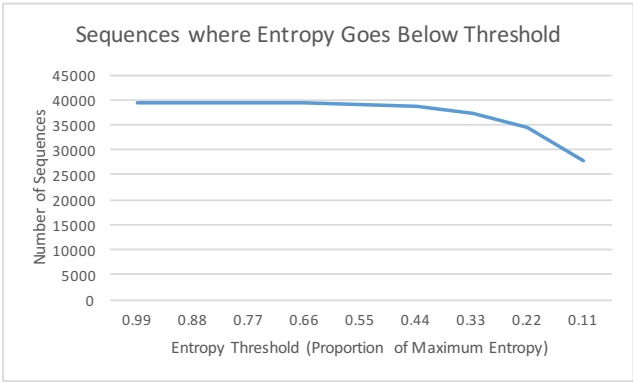

Figure 5: Sequences Crossing Entropy Thresholds

6 CONCLUSION AND FUTURE WORK

We presented two models for the real-time, early prediction of shopping session outcomes on an e-commerce platform. We demonstrated that LSTM RNNs generalize better and with less data than high-order Markov chain models used in previous work. Our approach, in addition to distinguishing *browsing-only* and *cart-interaction* sessions, can also accurately discriminate between cart abandonment and purchase sessions. Future work will focus on features, single RNN architectures, and decision strategies.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. (2015). http://tensorflow.org/ Software available from tensorflow.org.

[2] Aditya Awalkar, Ibrahim Ahmed, and Tejas Nevrekar. 2016. Prediction of User's Purchases using Clickstream Data. *International Journal of Engineering Science and Computing* (2016).

[3] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A Neural Probabilistic Language Model. *J. Mach. Learn. Res.* 3 (March 2003), 1137–1155.

[4] Dimitris Bertsimas, Adam J. Mersereau, and Nitin R. Patel. 2003. Dynamic Classification of Online Customers. In *Proceedings of the Third SIAM International Conference on Data Mining, San Francisco, CA, USA, May 1-3, 2003*. 107–118.

[5] Robert Florentin. 2016. *Online shopping abandonment rate a new perspective : the role of choice conflicts as a factor of online shopping abandonment.* Master's thesis. University of Twente.

[6] Joshua T. Goodman. 2001. A Bit of Progress in Language Modeling. *Comput. Speech Lang.* 15, 4 (Oct. 2001), 403–434.

[7] Bo-June Paul Hsu and James R. Glass. 2008. Iterative language model estimation: efficient data structure & algorithms. In *INTERSPEECH 2008, 9th Annual Conference of the International Speech Communication Association, Brisbane, Australia, September 22-26, 2008*. 841–844.

[8] Monika Kukar-Kinney and Angeline G. Close. 2010. The determinants of consumers' online shopping cart abandonment. *Journal of the Academy of Marketing Science* 38, 2 (2010), 240–250.

[9] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *The 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010)*.

[10] Deepak Pai, Abhijit Sharang, Meghanath Macha Yadagiri, and Shradha Agrawal. 2014. *Modelling Visit Similarity Using Click-Stream Data: A Supervised Approach.* Springer International Publishing, 135–145.

[11] The Statistics Portal. 2014. Digital buyer penetration worldwide from 2014 to 2019. http://www.statista.com/statistics/261676/digital-buyer-penetration-worldwide/. (2014). Accessed: 2016-09-10.

[12] Armando Vieira. 2015. Predicting online user behaviour using deep learning algorithms. *The Computing Research Repository (CoRR)* abs/1511.06247 (2015).

[13] Zhenzhou Wu, Bao Hong Tan, Rubing Duan, Yong Liu, and Rick Siow Mong Goh. 2015. Neural Modeling of Buying Behaviour for E-Commerce from Clicking Patterns. In *Proceedings of the 2015 International ACM Recommender Systems Challenge (RecSys '15 Challenge)*. ACM, New York, NY, USA, Article 12, 4 pages.

[14] Zhengzheng Xing, Jian Pei, and Eamonn Keogh. 2010. A Brief Survey on Sequence Classification. *SIGKDD Explor. Newsl.* 12, 1 (Nov. 2010), 40–48.

[15] Yin Xu and Jin-Song Huang. 2015. Factors Influencing Cart Abandonment in the Online Shopping Process. *Social Behavior and Personality: an international journal* 43, 10 (Nov. 2015).

[16] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent Neural Network Regularization. *The Computing Research Repository (CoRR)* abs/1409.2329 (2014). http://arxiv.org/abs/1409.2329