Understanding Ecommerce Clickstreams: A Tale of Two States

Humphrey Sheil
Cardiff University
Cardiff, Wales
sheilh@cardiff.ac.uk

Omer Rana
Cardiff University
Cardiff, Wales
RanaOF@cardiff.ac.uk

Ronan Reilly
Maynooth University
Maynooth, Ireland
Ronan.Reilly@mu.ie

ABSTRACT

We present an analysis of Ecommerce clickstream data using Recurrent Neural Networks (RNN), Gated Recurrent Units (GRU) and Long-Short Term Memory (LSTM). Our analysis highlights the substantial difference in the predictive power of LSTM models depending on whether or not hidden state is shared across batches and also assesses the ability of RNNs to learn and use both session-local and dataset-global information under different sampling strategies. We propose random sampling combined with stateless LSTM for optimal performance of LSTM in an Ecommerce domain.

KEYWORDS

Deep Learning, Recurrent Neural Networks (RNN), Long Short Term Memory (LSTM), Clickstreams, Ecommerce

ACM Reference Format:
Humphrey Sheil, Omer Rana, and Ronan Reilly. 2018. Understanding Ecommerce Clickstreams: A Tale of Two States. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Recurrent Neural Networks (RNNs) are one of the most common forms of neural networks in use today. They are well-suited to sequence processing tasks such as language modelling, tagging, translation and image captioning [20].

In recent work [28], we applied multiple variants of the base RNN model to the problem of analysing user *clickstream* data in order to predict user intent in an Ecommerce setting. In that work we tuned standard RNN hyperparameters such as dropout, batch size, learning rate, optimiser selection, sampling strategy, number and size of layers and also tested and employed less widely-used techniques such as skip connections. However, implementing skip connections required us to apply fine-grained control over each layer in the model, in particular how hidden state is passed both between batches and between layers. We noticed that choosing whether or not to pass hidden state between batches had a substantial impact on model performance. In fact, deciding how to handle hidden state and order of presentation of examples to the model during training were the two most important design decisions contributing to the predictive power of the model.

Although RNNs have been used recently to process clickstream sequences, more traditional work has utilised Gradient Boosted Machines (GBM) [9] and Field-aware Factorisation Machines (FFM) [17] to perform this analysis. Both of these approaches work well, especially when domain and dataset-specific features are used. Typical features constructed provide global context to the model, e.g. frequency-based measures of item popularity. RNNs by contrast do not receive this information explicitly, but can accrue it over time and training epochs.

Virtually all Ecommerce systems can be thought of as a generator of clickstream data - a log of $\{item - userid - action\}$ tuples which captures user interactions with the system. A chronological set of these tuples grouped by user ID is commonly known as a **session**.

In an Ecommerce context, we can think of local state as the individual "story" for a single user - their clicks including dwelltime and items provide insight into their intent (buy vs browse). Additionally, we posit that there is a global state telling a second story about the dataset, over and above the first and most immediate story encoded in each individual user session. Examples of global state are specific items going on sale for a short period of time and seasonality of particular items over weeks and months. It is intuitively appealing to think of both global and local patterns encoded in clickstream data that can be parsed and understood by LSTM to improve predictive performance. However, the inability of LSTM to handle very long sequences (and thus learn global patterns) is also well-known [22]. Our goal is to examine how well LSTM can learn global state in an Ecommerce context.

Electing to pass hidden state between batches when training is, inasmuch as we can determine, the default setting for RNN word language models. RNN word language models are also frequently used as the starting point for new sequence processing models and applications. Colloquially, passing state is referred to as stateful LSTM while choosing not to pass state is known as stateless LSTM. However there do not appear to be any formal references to this model configuration in the literature, even as a tips and tricks entry.

Practitioners are also faced with another important decision - how to sample from the dataset at training time to maximise performance at inference time. In the Experiments section, we measure the performance of multiple RNN variants under different settings to propose the optimal RNN configuration for Ecommerce clickstream analysis.

Sampling and hidden state strategy are very important to overall model performance. One example of the difference in model predictive power when hidden layer state is either re-used or discarded between batches is shown in Figure 1. When it is discarded, our best LSTM-based model is able to recover over 98% of the performance of a strong baseline - the State of the Art (SotA) model for this task [25]. With re-use, the model is far less effective.

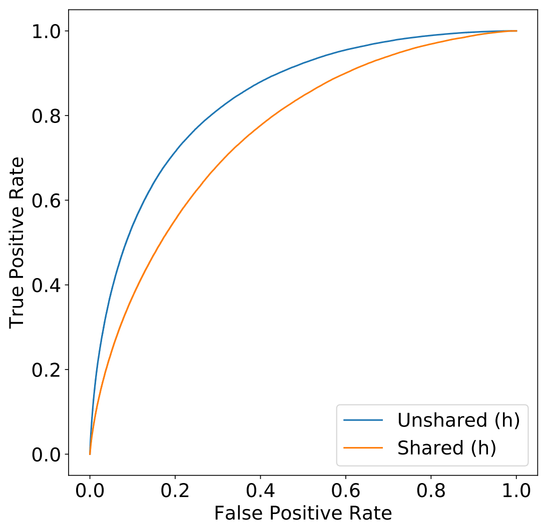

Figure 1: ROC curves for two LSTM models on the RecSys 2015 test set. Both models are identical apart from whether or not hidden state is re-used between batches during training. The "no sharing" model displays superior classification accuracy.

2 RELATED WORK

This section focuses on two sub-problems that meet in this paper: how to process and classify clickstreams effectively and the search for an optimal RNN model architecture for this task.

The problem of user intent or session classification in an online setting has been heavily studied, with a variety of classic machine learning and deep learning modelling techniques employed. [25] was the original winner of the competition using one of the the datasets considered here using a commercial implementation of GBM (a derivative has since been made publically available [8]) with extensive feature engineering and is still to our knowledge the SotA implementation for this dataset. The paper authors also made their model predictions freely available and we used these to compare our model performance to theirs.

[14] uses RNNs on a subset of the same dataset to predict the next session click (regardless of user intent) so removed 1-click sessions and merged clickers and buyers, whereas this work remains focused on the user intent classification problem. [21] compares [14] to a variety of classical Machine Learning algorithms on multiple datasets and finds that performance varies considerably by dataset. [33] extends [14] with a variant of LSTM to capture variations in dwelltime between user actions. User dwelltime is considered an important factor in multiple implementations and has been addressed in a variety of ways. For shopping behaviour prediction, [30] uses a mixture of Recurrent Neural Networks and treats the problem as a sequence-to-sequence translation problem, effectively combining two models (prediction and recommendation) into one. However only sessions of length 4 or greater are considered - removing the bulk from consideration. From [16], we know that short sessions are very common in Ecommerce datasets, moreover a user's most recent actions are often more important in deciphering their intent than older actions. Therefore we argue that all session lengths should be included.

Broadening our focus to include the general use of RNNs in the Ecommerce domain, Recurrent Recommender Networks are used in [32] to incorporate temporal features with user preferences to improve recommendations, to predict future behavioural directions, but not purchase intent. [29] further extends [14] by focusing on data augmentation and compensating for shifts in the underlying distribution of the data.

The search for better LSTM model architectures is almost as old as LSTM itself [10], [2]. In [13], the authors found that the forget gate and the output activation function are the most critical components in the LSTM cell. In [7], the authors found that more advanced recurrent units (i.e. variants such as LSTM and GRU that incorporate gating mechanisms) regularly outperform standard recurrent units and that for the most part, LSTM and GRU provide equivalent performance. [1] employs layer normalisation to reduce the training time for recurrent neural networks - an important goal given the inordinate time required to train SotA models on larger datasets. In [19], regularisation is applied to RNNs stochastically, with the aim of improving generalisation.

That the performance of neural network-based models depends heavily on the ordering of input sequences is well known. The authors of [5] argue that careful selection of training samples can achieve better generalisation. In [12], optimal performance on a sequence generation task is achieved by preserving order during training, while [4] observes that faster convergence can be observed if batch order is randomised between epochs. These observations are seemingly at odds with each other, and serve to illustrate the requirement for domain and dataset-specific tuning of training algorithms.

3 RECURRENT NEURAL NETWORKS

Recurrent neural networks [26] (RNNs) are a specialised class of neural networks for processing sequential data. A recurrent network is deep in *time* rather than space and arranges hidden state vectors h_t^l in a two-dimensional grid, where $t = 1 \ldots T$ represents time and $l = 1 \ldots L$ is the depth. All intermediate vectors h_t^l are computed as a function of h_{t-1}^l and h_t^{l-1}. Through these hidden vectors, each output y at some particular time step t becomes an approximating function of all input vectors up to that time, $x_1, \ldots, x_t$ [18].

3.0.1 LSTM and GRU. Long Short-Term Memory (LSTM) [15] is an extension to standard RNNs designed to address the twin problems of vanishing and exploding gradients during training [23]. Vanishing gradients make learning difficult as the correct (downward) trajectory of the gradient is difficult to discern, while exploding gradients make training unstable - both are undesirable outcomes. Long-term dependencies in the input data, causing a deep computational graph which must iterate over the data are the root cause of vanishing / exploding gradients. In [11], the authors explain this phenomenon succinctly. Like all deep learning models,

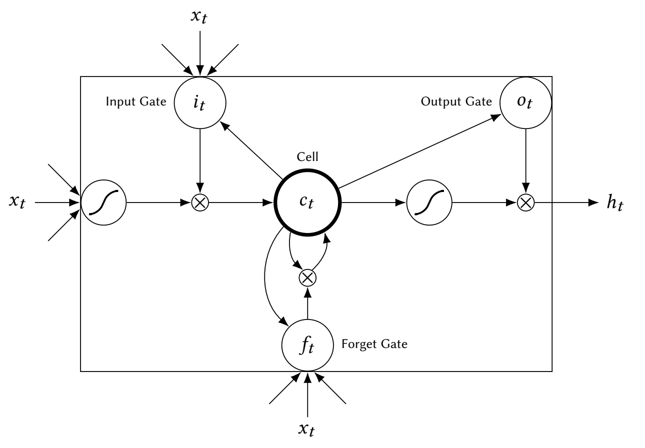

Figure 2: A single LSTM cell, depicting the hidden and cell states, as well as the three gates controlling memory (input, forget and output).

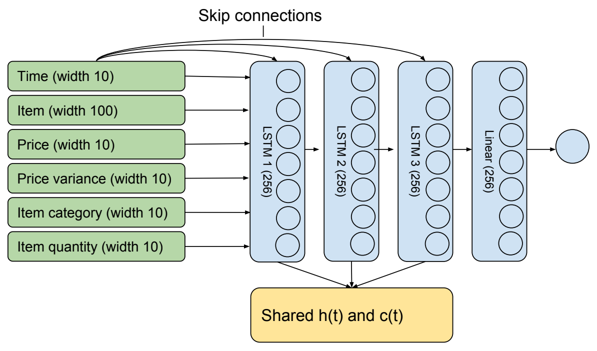

Figure 3: Schematic illustrating the general model architecture used for all experiments. The three most important variables are whether or not to re-use hidden state across batches, the recurrence cell type and training sampler used.

RNNs require multiplication by a matrix W. After t steps, this equates to multiplying by W^t. Therefore:

$$W^t = (V diag(\lambda) V^{-1})^t = V diag(\lambda)^t V^{-1} \tag{1}$$

Eigenvalues (λ) that are not more or less equal to 1 will either explode if they are > 1, or vanish if they are < 1. Gradients will then be scaled by $diag(\lambda)^t$.

LSTM solves this problem by using an internal recurrence, which stabilises the gradient flow, even over long sequences. However this comes at the price of complexity. For each element in the input sequence, each layer computes the following function:

$$\begin{aligned}
i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{(t-1)} + b_{hi}) \\
f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{(t-1)} + b_{hf}) \\
g_t &= \tanh(W_{ig}x_t + b_{ig} + W_{hc}h_{(t-1)} + b_{hg}) \\
o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{(t-1)} + b_{ho}) \\
c_t &= f_t * c_{(t-1)} + i_t * g_t \\
h_t &= o_t * \tanh(c_t)
\end{aligned} \tag{2}$$

where:
h_t is the hidden state at time t,
c_t is the cell state at time t,
x_t is the hidden state of the previous layer at time t or $input_t$ for the first layer,
i_t, f_t, g_t, o_t are the input, forget, cell, and out gates, respectively,
σ is the sigmoid function.

3.1 Model Structure / Hidden layers

In Deep Learning, the term "hidden layers" is used to refer to any model layer where the training data does not stipulate the desired output for these layers. Instead the training algorithm is free to use these layers to construct the best approximation for the desired function f^* - in other words hidden layers contribute to the overall model capacity in the form of learnable / trainable parameters.

Figure 3 below illustrates the standard model under discussion here - independent of the recurrent cell type used. The cells are organised into three layers with 256 cells per layer. The first layer accepts input from a group of input embeddings (unique input values are mapped to a corresponding vector of real values), while the output of the last layer is combined using a linear layer and passed through a non-linearity to generate a session prediction.

4 EXPERIMENTS

Our experiments focused on the two areas of RNN training where we observed significant differences in RNN performance depending on the choice taken:

- Whether or not hidden state was shared between batches.
- The sampling method used to construct batches as part of the training algorithm.

4.1 Desired Task

Predicting a users intent to purchase from their clickstream is a difficult task [27]. Clickers (users who only click and never purchase within a session) and buyers (users who click and also purchase at least one item within a single session) can appear to be very similar, right up until a purchase action occurs. Additionally, the ratio between clickers and buyers is always heavily imbalanced - and can be 20:1 in favour of clickers or higher. An uninterested user will often click on an item during browsing as there is no cost to doing so - an uninterested user will not *purchase* an item however. As noted in [30], shoppers behave differently when visiting online vs physical stores and online conversion rates are substantially lower, for a variety of reasons.

When a merchant has increased confidence that a subset of users are more likely to purchase, they can use this information in the form of preemptive actions to maximise conversion and yield. The merchant may offer a time-limited discount, spend more on targeted (and relevant) advertising to re-engage these users, create bundles of complementary products to push the user to complete their purchase, or even offer a lower-priced own-brand alternative if the product is deemed to be fungible.

4.2 Dataset Used

The RecSys 2015 Challenge [3] is a set of Ecommerce clickstreams well suited to testing purchase prediction models. It is reasonable in size, consisting of 9.2 million user sessions. These sessions are anonymous and consist of a chronological sequence of time-stamped events describing user interactions (clicks) with content while browsing and shopping online. The dataset also contains a very high proportion of short length sessions (<= 3 events), making this problem setting quite difficult for RNNs to solve.

No sessions were excluded - the dataset was used in its entirety. This means that for sequences with just one click, we require the trained embeddings to accurately describe the item, and time of viewing by the user to accurately classify the session, while for longer sessions, we can rely more on the RNN model to extract information from the sequence. This decision makes the training task harder for our RNN model, but is a fairer comparison to previous work using GBM where all session lengths were also included [25],[31],[27]. Lastly, the dataset is quite imbalanced - the class of interest (buyers) represents just 5% of the total number of samples.

4.3 RNNs vs GRU vs LSTM

Gated Recurrent Units, or GRU [6] are a simplification of LSTM, with one less gate and the hidden state and cell state vectors combined. These modifications mean that GRU is less computationally intensive to train versus LSTM. In practice, both LSTM and GRU are often used interchangeably and the performance difference between both cell types is often minimal and / or dataset-specific. RNN cells are the simplest of all recurrence cell types, with just a feedback loop from time $t-1$ to time t and possessing none of the gate structures / arrays used by GRU or LSTM.

Table 1 shows the impact of stateful vs stateless hidden state sharing on 4 widely-used RNN architectures.

Model	Stateful AUC	Stateless AUC
LSTM	0.75	0.839
GRU	0.756	0.831
RNN (TANH)	0.716	0.807
RNN (RELU)	0.789	0.826

Table 1: The effect of sharing hidden layer parameters on 4 widely used RNN model architectures - LSTM, GRU, and vanilla RNN using TANH or RELU non-linear activation functions. In all cases the effect of not using non-local state is a noticeable increase in AUC performance for a binary classification task. We use AUC to measure model performance since the classes are imbalanced.

4.4 Chronological Training Regime

A common technique to prevent over-fitting is to randomly sample from the training set during training and present disjoint samples to the model. However, if our goal is for LSTM to learn trends that develop over time then we must train chronologically and assume that there is a progression through time that our LSTM model can learn to discern between session outcomes. Therefore we re-order the training and testing set by time and perform both training and inference *chronologically*. As figure 4 shows, the initial results are very positive with a higher training performance, but roughly 20% into the epoch, training degrades substantially and does not recover. Validation performance is also significantly reduced. Therefore we conclude, that with conventional LSTM at least, there is no clear evolution in patterns over linear time that can be used to improve performance. Even though clickstreams are at some level a time series, the best model performance is achieved when *not* treating them as time series and instead sampling randomly. This finding is closely related to the initial finding shown in 1, where it is counter-productive to pass hidden state across batch boundaries.

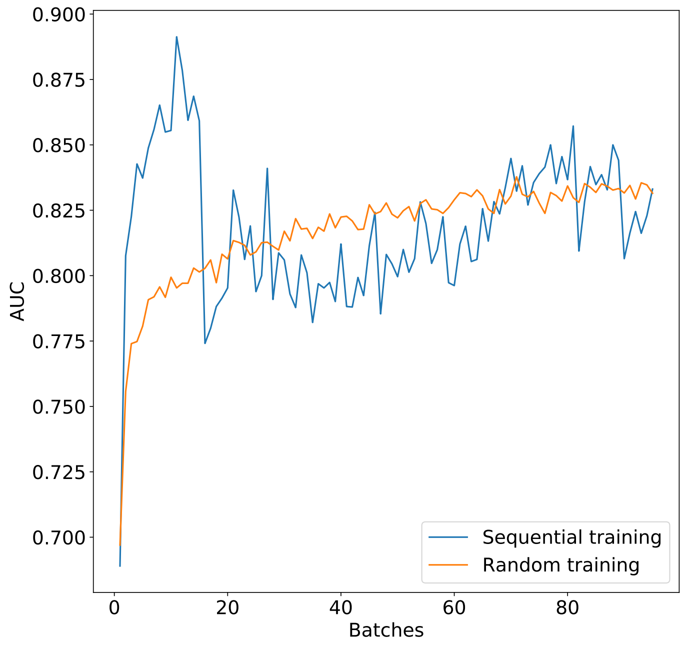

Figure 4: Training performance of LSTM under two sampling regimes: chronological (sequential) and random. Random presentation of examples performs better overall, both in training and at inference time on the test set.

4.5 Parameter Reduction

Our second experiment tests the theory that by selective ablation of certain parts of the model, we can remove the model's ability to pay attention to global data features. We reduce model capacity by freezing the embeddings only and retaining all of the RNN capacity. Our motivation in doing this is that not all model parameters are created equal - the embedding for an infrequently-encountered item will have far less effect on model performance than the hidden state contained directly in the model itself. Therefore we:

- Froze the entire embeddings layer - model loss was not back-propagated to the input layer.

- Moved from LSTM to vanilla RNN with RELU activations - removing internal memory from the model itself.

Under normal training conditions, the model loss is back-propagated all the way into the aggregate embedding layer, in effect treating these embeddings as a trainable memory for concepts such as popular and unpopular items / categories / dates, similar versus dissimilar item pairs and so on. With the embedding layers frozen, model performance does indeed reduce, as the figure below demonstrates.

These two changes resulted in a very large decrease in the number of the model parameters - from 6,945,871 to 367,873, or a reduction of 94%. As shown in table 2, these changes caused the model to under-perform our best model, but perhaps not by as much as expected.

Model	Num parameters	Test AUC
LSTM	6,945,871	0.839
LSTM	1,470,721	0.74
RNN (TANH)	367,873	0.722
RNN (RELU)	367,873	0.762

Table 2: The effect of removing the ability of RNN and LSTM to store dataset statistics other than those directly obtainable from a session (i.e. local). Due to the restrictions imposed, both RNN models have 5% of the best model parameters yet are able to recover 86% and 90% of the predictive power.

4.6 Implementation

The experiments were carried out using PyTorch [24]. In the PyTorch framework, the chronological and random training regime was implemented using the `SequentialSampler` and `RandomSampler` classes, both sub-classes of the `Sampler` class. Hidden state was initialised as a zero tensor in all cases and then either re-zeroed between batches (to complement random presentation of samples during training), or re-used between batches in the chronological training case. When training chronologically, the hidden state was detached from the computational graph after each batch to avoid the automatic differentiation from back-propagating all the way back to the beginning of each epoch.

5 SUMMARY AND CONCLUSION

We presented a series of experiments using different Recurrent Neural Network types to investigate if both global and local (session-level) patterns in Ecommerce datasets are used to infer user intent. The conclusion is yes, albeit with some caveats: although training results were substantially affected when the training set-up was configured to enable global training, results on the validation and test sets under-performed a training configuration where no global patterns or context were assumed. Instead, RNNs can use trainable embeddings to learn global statistics over time to improve performance.

Our results demonstrate that careful selection and configuration of both model and training regime is necessary when applying RNNs / LSTM to the domain of Ecommerce and clickstream analysis. Moreover, currently accepted best practices for RNN word language models do not automatically transfer to clickstream analysis and should be tested carefully on new datasets and domains.

In future work, we plan to investigate if the failure of a chronological training regime for LSTM is caused by an inherent weakness of the LSTM architecture, or if better training regimes or modifications can be found.

REFERENCES

[1] Lei Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer Normalization. *CoRR* abs/1607.06450 (2016). arXiv:1607.06450 http://arxiv.org/abs/1607.06450

[2] Justin Bayer, Daan Wierstra, Julian Togelius, and Jürgen Schmidhuber. 2009. Evolving Memory Cell Structures for Sequence Learning. In *Artificial Neural Networks – ICANN 2009*, Cesare Alippi, Marios Polycarpou, Christos Panayiotou, and Georgios Ellinas (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 755–764.

[3] David Ben-Shimon, Alexander Tsikinovsky, Michael Friedmann, Bracha Shapira, Lior Rokach, and Johannes Hoerle. 2015. RecSys Challenge 2015 and the YOOCHOOSE Dataset. In *Proceedings of the 9th ACM Conference on Recommender Systems (RecSys '15)*. ACM, New York, NY, USA, 357–358. https://doi.org/10.1145/2792838.2798723

[4] Yoshua Bengio. 2012. Practical recommendations for gradient-based training of deep architectures. *CoRR* abs/1206.5533 (2012). arXiv:1206.5533 http://arxiv.org/abs/1206.5533

[5] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum Learning. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML '09)*. ACM, New York, NY, USA, 41–48. https://doi.org/10.1145/1553374.1553380

[6] KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. *CoRR* abs/1409.1259 (2014). arXiv:1409.1259 http://arxiv.org/abs/1409.1259

[7] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *CoRR* abs/1412.3555 (2014). arXiv:1412.3555 http://arxiv.org/abs/1412.3555

[8] Anna Veronika Dorogush, Andrey Gulin, Gleb Gusev, Nikita Kazeev, Liudmila Ostroumova Prokhorenkova, and Aleksandr Vorobev. 2017. Fighting biases with dynamic boosting. *CoRR* abs/1706.09516 (2017). arXiv:1706.09516 http://arxiv.org/abs/1706.09516

[9] Jerome H. Friedman. 2000. Greedy Function Approximation: A Gradient Boosting Machine. *Annals of Statistics* 29 (2000), 1189–1232.

[10] Felix A. Gers and Juergen Schmidhuber. 2000. *Recurrent Nets That Time and Count.* Technical Report.

[11] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning.* MIT Press. http://www.deeplearningbook.org.

[12] Alex Graves. 2013. Generating Sequences With Recurrent Neural Networks. *CoRR* abs/1308.0850 (2013). arXiv:1308.0850 http://arxiv.org/abs/1308.0850

[13] Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R. Steunebrink, and Jürgen Schmidhuber. 2015. LSTM: A Search Space Odyssey. *CoRR* abs/1503.04069 (2015). arXiv:1503.04069 http://arxiv.org/abs/1503.04069

[14] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based Recommendations with Recurrent Neural Networks. *CoRR* abs/1511.06939 (2015). arXiv:1511.06939 http://arxiv.org/abs/1511.06939

[15] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* 9, 8 (Nov. 1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[16] Dietmar Jannach, Malte Ludewig, and Lukas Lerche. 2017. Session-based item recommendation in e-commerce: on short-term intents, reminders, trends and discounts. *User Modeling and User-Adapted Interaction* 27 (2017), 351–392.

[17] Yuchin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. 2016. Field-aware Factorization Machines for CTR Prediction. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16)*. ACM, New York, NY, USA, 43–50. https://doi.org/10.1145/2959100.2959134

[18] Andrej Karpathy, Justin Johnson, and Fei-Fei Li. 2015. Visualizing and Understanding Recurrent Networks. *CoRR* abs/1506.02078 (2015). arXiv:1506.02078 http://arxiv.org/abs/1506.02078

[19] David Krueger, Tegan Maharaj, János Kramár, Mohammad Pezeshki, Nicolas Ballas, Nan Rosemary Ke, Anirudh Goyal, Yoshua Bengio, Hugo Larochelle, Aaron C. Courville, and Chris Pal. 2016. Zoneout: Regularizing RNNs by Randomly Preserving Hidden Activations. *CoRR* abs/1606.01305 (2016). arXiv:1606.01305 http://arxiv.org/abs/1606.01305

[20] Zachary Chase Lipton. 2015. A Critical Review of Recurrent Neural Networks for Sequence Learning. *CoRR* abs/1506.00019 (2015). arXiv:1506.00019 http://arxiv.org/abs/1506.00019

[21] Malte Ludewig and Dietmar Jannach. 2018. Evaluation of Session-based Recommendation Algorithms. *CoRR* abs/1803.09587 (2018). arXiv:1803.09587

http://arxiv.org/abs/1803.09587

[22] Daniel Neil, Michael Pfeiffer, and Shih-Chii Liu. 2016. Phased LSTM: Accelerating Recurrent Network Training for Long or Event-based Sequences. *CoRR* abs/1610.09513 (2016). arXiv:1610.09513 http://arxiv.org/abs/1610.09513

[23] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the Difficulty of Training Recurrent Neural Networks. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28 (ICML'13)*. JMLR.org, III–1310–III–1318. http://dl.acm.org/citation.cfm?id=3042817.3043083

[24] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. (2017).

[25] Peter Romov and Evgeny Sokolov. 2015. RecSys Challenge 2015: Ensemble Learning with Categorical Features. In *Proceedings of the 2015 International ACM Recommender Systems Challenge (RecSys '15 Challenge)*. ACM, New York, NY, USA, Article 1, 4 pages. https://doi.org/10.1145/2813448.2813510

[26] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. 1986. Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1. MIT Press, Cambridge, MA, USA, Chapter Learning Internal Representations by Error Propagation, 318–362. http://dl.acm.org/citation.cfm?id=104279.104293

[27] Humphrey Sheil and Omer Rana. 2017. Classifying and Recommending Using Gradient Boosted Machines and Vector Space Models. In *Advances in Computational Intelligence Systems. UKCI 2017.*, Zhang Q Chao F., Schockaert S. (Ed.), Vol. 650. Springer, Cham. https://doi.org/10.1007/978-3-319-66939-7_18

[28] Humphrey Sheil, Omer Rana, and Ronan Reilly. 2018. Predicting purchasing intent: Automatic Feature Learning usingRecurrent Neural Networks. In *ACM SIGIR Forum*. ACM.

[29] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved Recurrent Neural Networks for Session-based Recommendations. *CoRR* abs/1606.08117 (2016). arXiv:1606.08117 http://arxiv.org/abs/1606.08117

[30] Arthur Toth, Louis Tan, Giuseppe Di Fabbrizio, and Ankur Datta. 2017. Predicting Shopping Behavior with Mixture of RNNs. In *ACM SIGIR Forum*. ACM.

[31] Maksims Volkovs. 2015. Two-Stage Approach to Item Recommendation from User Sessions. In *Proceedings of the 2015 International ACM Recommender Systems Challenge (RecSys '15 Challenge)*. ACM, New York, NY, USA, Article 3, 4 pages. https://doi.org/10.1145/2813448.2813512

[32] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J. Smola, and How Jing. 2017. Recurrent Recommender Networks. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining (WSDM '17)*. ACM, New York, NY, USA, 495–503. https://doi.org/10.1145/3018661.3018689

[33] Yu Zhu, Hao Li, Yikang Liao, Beidou Wang, Ziyu Guan, Haifeng Liu, and Deng Cai. 2017. What to Do Next: Modeling User Behaviors by Time-LSTM. In *IJCAI*.