Understanding Consumer Behavior with Recurrent Neural Networks

Tobias Lang, Matthias Rettenmeier*

Abstract

Consumer behavior in e-commerce can be described by sequences of interactions with a webshop. We show that recurrent neural networks (RNNs) are a natural fit for modeling and predicting consumer behavior. In multiple aspects, RNNs offer advantages over existing methods that are relevant for real-world production systems. Applying RNNs directly to sequences of consumer actions yields the same or higher prediction accuracy than vector-based methods like logistic regression. Unlike the latter, the application of RNNs comes without the need for extensive feature engineering. In addition, we show that RNNs help us link individual actions directly to predictions in an intuitive way. This allows us to understand the implications consumer actions have on predicted probabilities over the course of the consumer's history. We demonstrate the advantages of RNNs on the empirical data of a large European online fashion platform.

1 Introduction

Predicting future consumer behavior is fundamental to many use-cases in e-commerce. Applications range from recommender systems over fraud detection to real-time bidding for online ad-inventory [4, 1, 15, 10, 2, 18]. Such predictions are based on indicators found in previous consumer behavior. For example, the time since a consumer last visited the webshop, the products that were looked at or the amount of products added to the cart. Behavior is captured in consumer histories, which are, in their raw form, sequences of interactions with the webshop. Interactions are of a particular type, they have a timestamp and additional information such as product details. The type is derived from the associated action, such as a product-view or a cart-addition, the timestamp is simply the time of the action (Fig.1).

Most of the popular machine learning methods used in e-commerce, including logistic regression, neural networks, and random forests, employ vector-based models: they operate on feature vectors of fixed length as input [3]. To apply them to predict consumer behavior, one needs to convert consumer histories into fixed sets of features. These features are usually hand-crafted by domain experts and reflect the indicators mentioned above. Finding indicators and designing a reliable set of features is critical for the prediction accuracy. It requires many iterations of empirical experiments and involves time-intensive, tedious human work. Furthermore, it is difficult to explain the reasoning for the prediction outcome, in terms of individual consumer actions, from vector-based models [13]. Predictions that are hard to explain make it tricky to offer answers to consumers asking for transparency about how algorithms determine their individual user experience [5]. Furthermore, a better understanding of trained models helps webshop providers to improve their services [16].

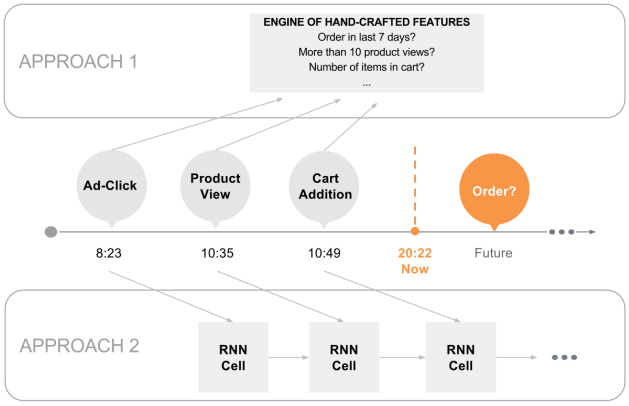

Figure 1: Two approaches to model consumer histories: vector-based methods (Approach 1) use hand-crafted non-sequential features to encapsulate the signals in the consumer history; RNNs (Approach 2) model the sequential consumer history directly.

In this paper, we show that recurrent neural networks (RNNs) are promising to overcome both shortcomings of vector-based methods, tedious feature engineering and lack of explainability. RNNs operate on sequences of varying length and therefore provide an appropriate match to consumer histories. We apply RNNs directly to series of captured consumer actions. RNNs maintain a latent state that is updated with each action. RNNs are trained to detect and preserve the predictive signals in the consumer histories. The latent state cor-

*Zalando adtech lab GmbH, Hamburg, Germany, {tobias.lang, matthias.rettenmeier}@zalando.de

responds to a representation of learned features; no further feature engineering is required.

RNNs allow to link individual actions to predictions in a straightforward manner. To our knowledge, in this work we provide the first visualizations that exploit this in the context of e-commerce. This allows to quantify how predictions are affected by specific actions or action sequences conducted by the consumer. These insights are drawn on a quantitative empirical basis—in contrast to vague intuitions that often drive product development in e-commerce.

We demonstrate the advantages of RNNs in experiments on large-scale real-world data-sets from Europe's leading online fashion platform Zalando[1], operating in multiple countries with millions of customers and a catalog comprising hundreds of thousands of products at any given moment. We focus on the example of predicting order probabilities, which is fundamental to many e-commerce and recommender system scenarios.

To summarize, our contributions are the following: (i) we show how consumer behavior can be predicted without sophisticated feature engineering by using RNNs; (ii) we provide an empirical comparison of prediction performance on real-world e-commerce data; and (iii) we demonstrate how RNNs are helpful in explaining the predictions for individual consumers.

In the next section, we provide background on common approaches applied in e-commerce today, in particular the utilization of vector-based methods. In Sec. 3, we give a short introduction to RNNs and show how they naturally fit the use-case of predicting future consumer behavior in e-commerce. In Sec. 4, we provide empirical results to compare the performance of RNNs with several baseline methods. In Sec. 5, we show how the predictions of RNNs can be explained in terms of individual consumer actions. We discuss related work in Sec. 6, before we conclude.

2 Background on vector-based methods and feature engineering

Vector-based machine learning methods like logistic regression take vectors $f = (f_1, \ldots, f_n)$ of fixed length n as inputs. Applying these methods on consumer histories of arbitrary length requires feature engineering: a fixed set of identifiers f_i has to be designed to capture the essence of an individual consumer history. Only signals that are encoded in the feature vector can be picked up by the prediction model.

Defining expressive features often requires both, domain knowledge as well as data-science intuition. In our example case of order prediction, one might have a sense of what indicators for an imminent purchase might be and how they could be conceived into features, like: Has the consumer added new products to the cart recently? Collecting these ideas for features in respect to the given prediction problem is just the first step in feature engineering.

Additional feature processing steps are often necessary to improve model performance. Preprocessing may be required if input features consist of numerical, ordinal, and categorical features at the same time; if features have different value ranges; or to ensure model robustness with model regularization. A common preprocessing approach is to create binary input vectors from the original input features [3]. For numerical features such a transformation is often done by assigning binary values to value ranges of the numerical feature. For example, a feature for order counts could be converted into buckets for zero orders, one order, two-five orders, and six plus orders.

While preprocessing is an important tool to improve model performance, it artificially increases the dimensionality of the input-vector. Also, the resulting binary features can be strongly correlated. Both outcomes make it difficult to tell which action patterns in the underlying consumer histories have a strong impact on the prediction outcome [13, 6]. We will discuss this in more detail in Sec. 5.

The specific set of features and their preprocessing have decisive effects on model performance. These effects can only be determined in experiments on historical data and in A/B tests. This makes feature engineering a critical, but time-consuming and tedious effort.

3 Predicting future consumer behavior with RNNs

In contrast to vector-based methods, recurrent neural networks (RNNs) take sequences $X = (x_1, \ldots, x_T)$ of varying length T directly as inputs. RNNs are built as connected sequences of computational cells. The cell at step t takes input x_t and maintains a hidden state $h_t \in \mathbb{R}^d$. This hidden state is computed from the input x_t and the cell state at the previous time-step h_{t-1} as

$$h_t = \sigma(W_x x_t + W_h h_{t-1} + b) \tag{3.1}$$

where W_x and W_h are learned weight matrices, b is a learned bias vector and σ is the sigmoid function. A hidden state h_t captures information from the input sequence $(x_1, \ldots, x_t)$ up to the current time-step t. Information from early inputs can thereby be preserved over time. The dimensionality d of the hidden state is a hyperparameter that is chosen according to the complexity of the temporal dynamics of the scenario.

In practice, more sophisticated computational cell

[1] http://www.zalando.com

types like long short-term memory cells (LSTMs) [8] are better in preserving long-term dependencies. LSTMs maintain an additional cell state C for long-term memory and calculate hidden and cell states h_t and C_t in the following cascade of gating operations:

$$
\begin{aligned}
f_t &= \sigma(W_f[h_{t-1}, x_t] + b_f) \\
i_t &= \sigma(W_i[h_{t-1}, x_t] + b_i) \\
\hat{C}_t &= \tanh(W_c[h_{t-1}, x_t] + b_C) \\
C_t &= f_t C_{t-1} + i_t \hat{C}_t \\
o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o) \\
h_t &= o_t \tanh(C_t)
\end{aligned}
$$

where W and b are learned weight matrices and bias vectors. The final hidden state h_T of an RNN can be used to classify a sequence: h_T is input into a prediction network, which can be a simple linear layer or a sequence of non-linear layers.

During training, the parameters W and b of the computational cells are adapted to detect signals in the input sequences that increase prediction accuracy. Input sequences X are compressed by this process into suitable feature vectors h_T. Often the compression process is viewed as feature learning from raw inputs and is the reason why work-intensive human feature engineering is not required before applying RNNs. The complexity of RNN models, however, yields longer processing time for learning and predicting when compared with vector-based methods. Model tuning RNNs can also be more complex for the same reason: there are more architectural choices and hyperparameters to tune. Nevertheless, we show here that even simple RNN architectures out-perform vector-based approaches.

3.1 Event-stream RNNs We propose to model the behavior of consumers with RNNs. Consumer histories are inherently sequential and of varying lengths T, making RNNs a natural model choice. In e-commerce, available data sources and prediction scenarios often change, making the generality of RNNs appealing as no problem-specific feature engineering has to take place.

RNNs can be applied to predict future consumer behavior in regression and classification settings, for example to predict interest in fashion brands or consumer life-time value. We focus on predicting the probablity $P(o^u \mid x_1^u, \ldots, x_T^u)$ of a consumer u to place an order o^u, which we model as a binary classification problem. For instance, we could be interested in orders in general or of specific products. The resulting probability estimates can be used in a recommender system to deliver appropriate product recommendations and webshop contents.

Consumer events x have an action type (product-views, cart-additions, orders, etc.), a timestamp and potentially additional information, for example details about the product that the consumer interacted with. We input sequences $X = (x_1, \ldots, x_T)$ of consumer events x directly into RNNs. During training, the RNN needs to learn to detect relevant action patterns in event streams. For instance, it may learn that a sequence of product views in the morning and a subsequent cart-addition in the evening increases the order probability more than the reversed sequence, in which a product is first placed into the cart before the consumer continues to view other products later.

3.2 Session-stream RNNs Recognizing relevant patterns in long input streams (> 100 actions) can turn out to be difficult for the human mind. To achieve better explainability, in many e-commerce applications consumer behavior can be viewed on the level of sessions. A session is a well-defined visit of a consumer to a webshop: a subsequence of events within the consumer's history that lay no further appart than a predefined time difference. Here, we split sequences of events into different sessions if there is a time-gap of more than 30 minutes in-between subsequent events.

We propose session streams $S = (s_1, \ldots, s_T)$ as an alternative input to RNNs to model consumer behavior and to predict orders as $P(o^u \mid s_1^u, \ldots, s_T^u)$. In this representation, the input s_t at time-step t describes a full session, compressing all its events into a single input vector by using a mild form of feature engineering on a session level. We use the following session features: a binary vector to indicate which action types occurred; the number of total events and the session length in seconds; and if available, the number of advertisement interactions between the current and the previous session (which are not part of the webshop session). Apart from explainability, session streams have a technical advantage over event streams: the required history horizons are shorter as multiple events are processed at the same time-step.

4 Experiments

We evaluate RNNs as a model for consumer behavior on the basis of the model's ability to estimate order probabilities. We focus on the probability that a consumer will order within a short term after she starts a session at a webshop. The experiments are run on large-scale data-sets containing millions of consumer histories at Europe's leading online fashion platform Zalando in two countries (exact numbers cannot be disclosed for business reasons). Consumer histories are compiled to event sequences from 19 different action types (product-view, order, cart-addition, etc.). Session starts of consumers are used as test points. For a

given session start, the task is to predict the probability that the consumer orders within the current or any following session within the next seven days. While in practice we apply our RNN approach mostly to predict product-specific orders, we focus on general orders here for simplicity. The consumer's events prior to the current session, potentially dating back many weeks, together with their timestamps constitute the input data. Apart from the start timestamp, no information concerning the current session is used as input. Sessions from six subsequent weeks in spring 2016 are used in the experiments, involving consumers with at least 15 previous actions (motivated by our practical use case). Sessions in the first 4 weeks are used for training, sessions in the last 2 weeks for testing, resulting in several million training and test sessions. 25% of the training sessions are used as a validation set. The two classes are highly unbalanced with more negatives (sessions without subsequent order) than positives.

4.1 RNN details We use a simple RNN architecture with a single LSTM layer and ten-dimensional cell states. The hidden state at the last time-step is combined with binary non-history features to make the final prediction in a logistic layer. Thus, the final prediction of the RNN is linear in the learned and non-history features. The non-history features describe time, weekday, and behavioral gender and are also provided to the baseline methods. Instead of absolute timestamps, the time differences $\Delta(x_{t-1}, x_t)$ to the previous inputs x_{t-1} are fed to the RNN at each time-step t. Furthermore, the difference between the last event x_T and the prediction time (the session start) is provided to the final prediction layer. For event-stream RNNs, history inputs $x_t \in \mathbb{R}^{20}$ consist of a one-hot encoding of the action type and the time difference. For session-stream RNNs, history inputs $s_t \in \mathbb{R}^{23}$ represent sessions with binary indicators which action types occurred, the time difference to the previous session and the characteristics described in Sec. 3.2. Time differences and, in case of session-stream RNNs, the total session event counts are logarithmized.

Different RNN models for event-stream and session-stream inputs (Sec. 3) are trained in each country. For event-stream RNNs, up to the last 200 consumers events are used as consumer history; for session-stream RNNs, up to the last 30 sessions. RNNs were implemented in Torch 7 using the package rnn [12]. The Adam optimizer with standard settings was used for training. A learning rate of 0.001 and Xavier initialization was chosen according to a small hyperparameter search using the validation data. Training from scratch took two hours for session-stream RNNs and about a day for event-stream RNNs. Since classes are highly unbalanced, we experimented with pre-training on re-sampled data-sets, but did not find significant improvements.

	Country 1		Country 2	
	NLL	AUC	NLL	AUC
Logistic regression with non-history features only	0.278	0.545	0.278	0.552
Logistic regression with all hand-crafted features	0.239	0.774	0.218	0.832
RNN with event stream	0.239	**0.778**	0.215	**0.843**
RNN with session stream	**0.238**	**0.778**	**0.213**	0.842

Table 1: Results for logistic regression and RNN models linear in the hand-engineered / learned features

	NLL	AUC
NN with hand-crafted features	0.214	0.841
RNN with session stream + non-linear pred.	**0.211**	**0.845**

Table 2: Results in Country 2 for a neural net and a RNN model non-linear in the hand-engineered / learned features

The RNNs used in the experiments are comparable to logistic regression: both models make linear predictions in a single logistic layer using a set of features which are either learned or hand-crafted. As an extension, we briefly explore an RNN with non-linear final prediction including an additional hidden layer of size 100 with ReLU activation function.

4.2 Baselines RNNs are compared to the common approach of using logistic regression with binary hand-engineered features [3] (Sec. 2). Logistic regression is vector-based and cannot model sequential consumer histories directly, but relies on well-chosen features. The features were determined in intensive feature engineering efforts over multiple months and have been used in production systems, resulting in several hundred binary features. Preprocessing steps for binarization were fine-tuned empirically. Most features describe the consumer history (see Table 3 for examples). In addition, the same non-history features as for the RNN were used. While logistic regression is linear in the hand-crafted features, it is non-linear in the consumer histories, due to the hand-engineered feature definitions. As an extension, we explore a neural net (NN) as a second vector-based model; it uses a hidden layer of size 750 and ReLU activation function; both parameters were optimized on the validation set. The Adam optimizer was used for training. Both baselines were implemented in Torch 7. Training took about 20 minutes for logistic regression and a few hours for the neural net.

4.3 Results All models are trained to minimize negative log-likelihood (NLL). As probability estimates are

required directly in many practical applications, we use NLL also for evaluation. In some applications, the resulting ranking of consumers is more important than the probabilities themselves. For this reason, we also report the area under the ROC curve (AUC).

Table 1 presents the results. The RNN models perform about as good as or better than logistic regression. Session-stream RNNs yield slightly higher accuracy than event-stream RNNs. Results for logistic regression employing only the non-history features show that most of the relevant information for order prediction is contained in the consumer histories.

Both models, logistic regression and the simple RNN, are linear in the hand-engineered / learned features. While more sophisticated non-linear models are beyond the scope of this paper, we briefly explored the potential of a non-linear extension to our model. For the larger Country 2 data-set we tested a neural net and a session-stream RNN with a two-layer prediction network. The results are shown in Table 2. Both achieve better performance than their linear counterparts, with the extended RNN achieving overall best performance.

4.4 Discussion The results for the baseline models show that the predictive signal is mostly contained in the consumer histories. The carefully hand-engineered features do capture this signal to a great extent. Nevertheless, the RNNs have learned to detect this signal without advanced feature engineering. RNNs achieve the same or higher prediction accuracy than both, the logistic regression and neural nets. In further results not reported here, we found that using hand-engineered features with other vector-based methods, including boosting techniques, did not lead to superior performance, either, indicating the quality and the limitation of our feature engineering. While this is a promising result for RNNs, we suspect further performance gains with RNNs are possible when employing more sophisticated RNN architectures. The slightly higher accuracy for the session-stream RNNs might be due to insufficient parameter optimization for the event-stream RNNs.

5 Explaining predictions

As machine learning models become ubiquituous in our everyday lives, demand for explaining their predictions is growing [5, 16, 14]. In the context of behavior prediction, we want to understand how previous consumer actions influence model predictions: How does order probability change when products are put into the cart? Does it decrease significantly if a consumer does not return to a webshop for two days? Answers to these questions are consumer-specific; they depend on the complete consumer history. RNNs provide the explanation by directly linking consumer actions with the model predictions. In contrast and contrary to popular beliefs, it is not straightforward to answer such questions with vector-based methods like logistic regression [13].

5.1 Vector-based methods Vector-based methods like logistic regression rely on hand-crafted features. During feature engineering, ordering and timing of consumer events are typically lost. This makes it difficult to relate individual actions in a consumer history to predictions without additional investigation.

In principle, one could evaluate the logistic regression model at every single time-step in the consumer history to determine the influence of individual events. However, this would involve the inefficient process of recalculating features for every time-step. Calculations at timesteps t and $t-1$ would be highly redundant: features at t represent the complete history until t and not only what happened inbetween $t-1$ and t.

Generally speaking, explaining the predictions of vector-based methods is more difficult than often assumed. This holds even for linear models like logistic regression. Features are often preprocessed, for example to binarize counts (Sec. 2). Furthermore, they are typically strongly correlated, making it troublesome to interpret individual coefficients [6]. Table 3 shows exemplary features weights in a logistic regression model used to predict order probabilities. If hundreds of features are utilized and are correlated and preprocessed, explaining the impact of consumer actions becomes a complex and confusing task [13].

5.2 RNNs RNNs link events over the course of consumer histories with predicted probabilities in a straightforward way, due to their natural modeling of sequences. Hidden states in the RNN are updated after each event. The hidden state at a specific timestep in combination with non-history features is all that is needed to make a prediction based on the consumer history until this point. Hidden states (and cell states for LSTM units) are calculated for all time-steps when making a prediction for a complete consumer history. In turn, all intermediate predictions come for free and without redundant calculations. This makes it straightforward to visualize the course of predicted probabilities in regard to the consumer events and without the requirement to interpret the cell states.

Visualizing RNN predictions on a consumer's event stream helps to understand which actions have a strong influence on the overall prediction. Fig. 2 shows the latent cell states and the corresponding predictions for the event streams of two consumers used in the experiments. While the cell values cannot be interpreted by themselves, we can see how they change for the given

♯product-views = 0	0.03	♯sessions = 1	-1.32	Last session < 1 hour ago	1.23
♯product-views = 1	0.51	♯sessions ∈ [2, . . . , 5]	0.02	Last session 1-24 hours ago	0.73
♯product-views ∈ [2, . . . , 5]	1.82	♯sessions ∈ [6, . . . , 10]	1.04	Last session 2-7 days ago	-0.02
♯product-views ∈ [6, . . . , 10]	0.73	♯sessions ∈ [11, . . . , 20]	2.38	Last session 8-30 days ago	-0.32
♯product-views ∈ [11, . . . , 50]	1.51	♯sessions > 20	0.73	Last session > 30 days ago	-0.08
. . .	. . .	. . .	. . .	. . .	. . .

Table 3: Exemplary features with potential weights of a logistic regression model for order prediction

inputs over time. The RNN has learned to detect consumer behavior indicative of future orders. The progress of predicted probabilities shows how individual actions like cart-additions increase or decrease the order probability. For example, for the first consumer, the order towards the beginning as well as the long time difference of 8279 minutes (>5 days) towards the end both lead to sudden drops in predicted probability.

It can be difficult for humans to comprehend patterns on the level of individual actions. In e-commerce, often one wants to understand consumer behavior on the level of sessions instead. This can be achieved with the session-stream representation (see Sec.3) that links sessions to predictions. Fig. 3 shows latent states and predictions over the session journey of an exemplary consumer. For example, Session -7 ends with an order, resulting in lower future order probability. The subsequent sessions are small, but have cart-additions, leading to slow, but steady increase in order probability.

6 Related work

Traditional approaches in e-commerce are based on vector-based methods like logistic regression with feature engineering [3]. Deep learning approaches to predict consumer behavior have become popular in recent years, but mostly do not model sequential behavior explicitly. For example, deep learning has been applied to predict customer churn in mobile telecommunications [18, 2] by converting sequential consumer data like phone calls and expenses into image-like representations instead of using RNNs. Deep belief networks and autoencoders have been used as probabilistic generative processes to predict sessions with orders from engineered features exploiting product interactions of consumers [17]. Non-recurrent deep learning models were applied to learn abstract representations from product attributes to predict future sales [1].

RNNs have been investigated in e-commerce and recommender system scenarios, but not to model consumer behavior in terms of different actions. RNNs have been applied on the stream of ad-impressions shown to individual consumers on search result pages to predict click rates for ads [19]. Other approaches apply RNNs to item recommendation within a single session based on the items a consumer interacted with in that session [7]. In contrast to our approach, consumer behavior is not modeled across sessions and not in the form of different action types, for instance losing the semantic difference between cart-additions and product-views. RNNs have been used for natural language processing to predict purchases from the contents of twitter messages [10].

The interpretability of RNN models was studied recently [9, 11]. It was argued that vector-based models are often not more interpretable than deep learning models [13]. Explaining the predictions of RNNs has not received much attention yet [14].

7 Conclusions

We have proposed an approach to apply RNNs to predict future consumer behavior in e-commerce. Consumer behavior is inherently sequential which makes RNNs a perfect fit. We are employing RNNs in production now which offers lsignificant advantages over existing methods: reduced feature engineering; improved empirical performance; and better prediction explanations. In the future, predictions on the level of products and individual tastes will be in our focus, enabling sophisticated recommendation products. This will require richer input descriptions at individual time-steps. Likewise, more sophisticated RNN architectures will be promising future research.

Acknowledgements We thank Roland Vollgraf and Nikolay Jetchev at Zalando Research for inspiring discussions and valuable feedback.

References

[1] C. Bracher, S. Heinz, and R. Vollgraf. Fashion DNA: Merging content and sales data for recommendation and article mapping. In *Workshop Machine learning meets fashion*, KDD, 2016.

[2] F. Castanedo, G. Valverde, J. Zaratiegui, and A. Vazquez. Using deep learning to predict customer churn in a mobile telecommunication network. 2016. http://www.wiseathena.com/pdf/wa_dl.pdf. Accessed on 6 Dec 2016.

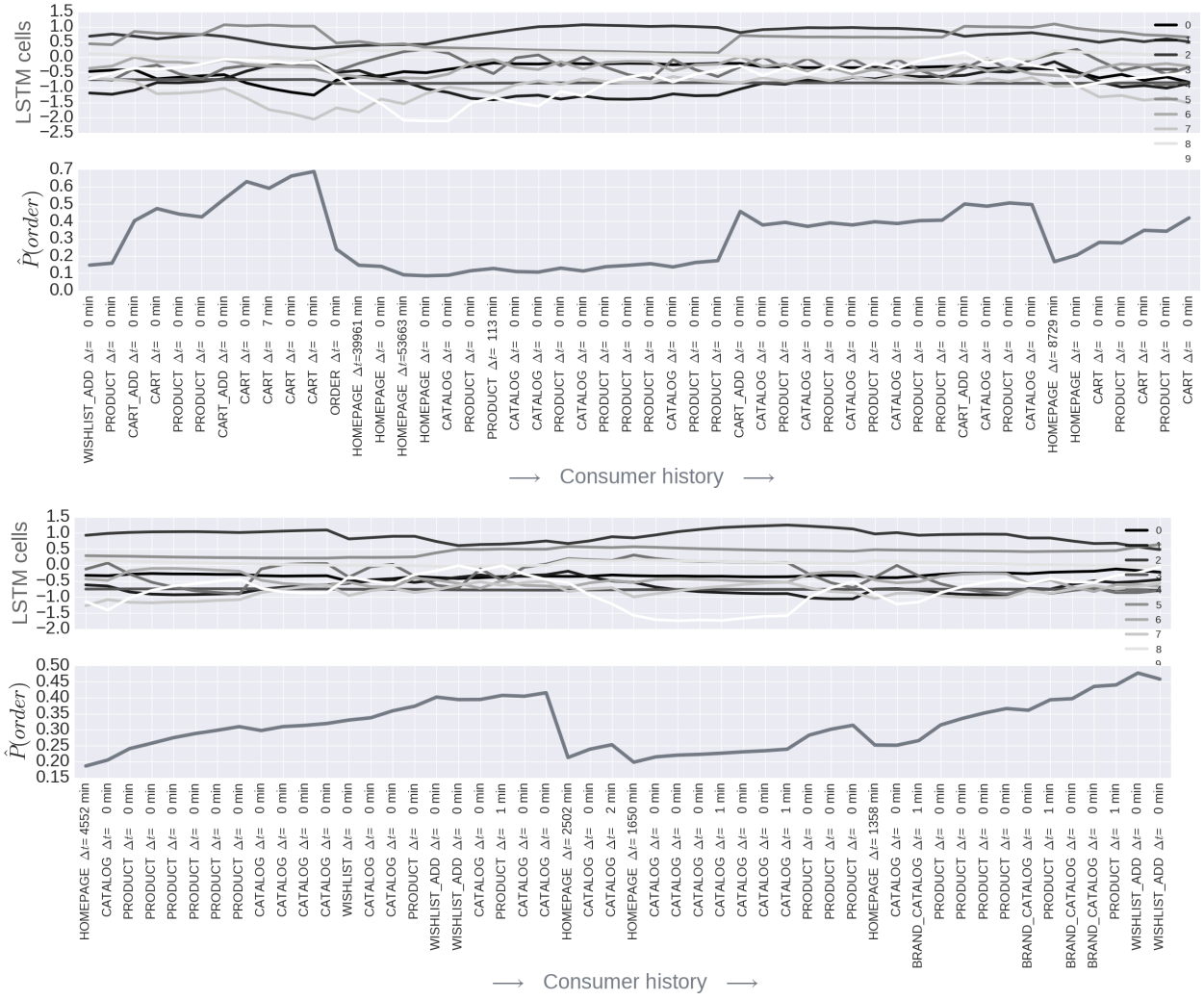

Figure 2: Event histories for two consumers are shown. We can explain the predictions of an event-stream RNN for each by linking their events with LSTM cells and predicted probabilities.

[3] O. Chapelle, E. Manavoglu, and R. Rosales. Simple and scalable response prediction for display advertising. *Transactions on Intelligent Systems and Technology*, 5, 2014.

[4] Y. Chen, P. Berkhin, B. Anderson, and N. Devanur. Real-time bidding algorithms for performance-based display ad allocation. In *Conf. on Knowledge Discovery and Data Mining (KDD)*, 2011.

[5] B. Goodman and S. Flaxman. European Union regulations on algorithmic decision-making and a 'right to explanation'. In *ICML Workshop on Human Interpretability in Machine Learning*, 2016.

[6] S. Haufe, F. Meinecke, K. Goergen, S. Daehne, J. Haynes, B. Blankertz, and F. Biessmann. On the interpretation of weight vectors of linear models in multivariate neuroimaging. *NeuroImage*, 87:96–110, 2014.

[7] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk. Session-based recommendations with recurrent neural networks. *Intern. Conf. on Learning Representations (ICLR)*, 2016.

[8] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[9] A. Karpathy, J. Johnson, and L. Fei-Fei. Visualizing and understanding recurrent networks. In *Intern. Conf. for Learning Representations Workshop Track*, 2016.

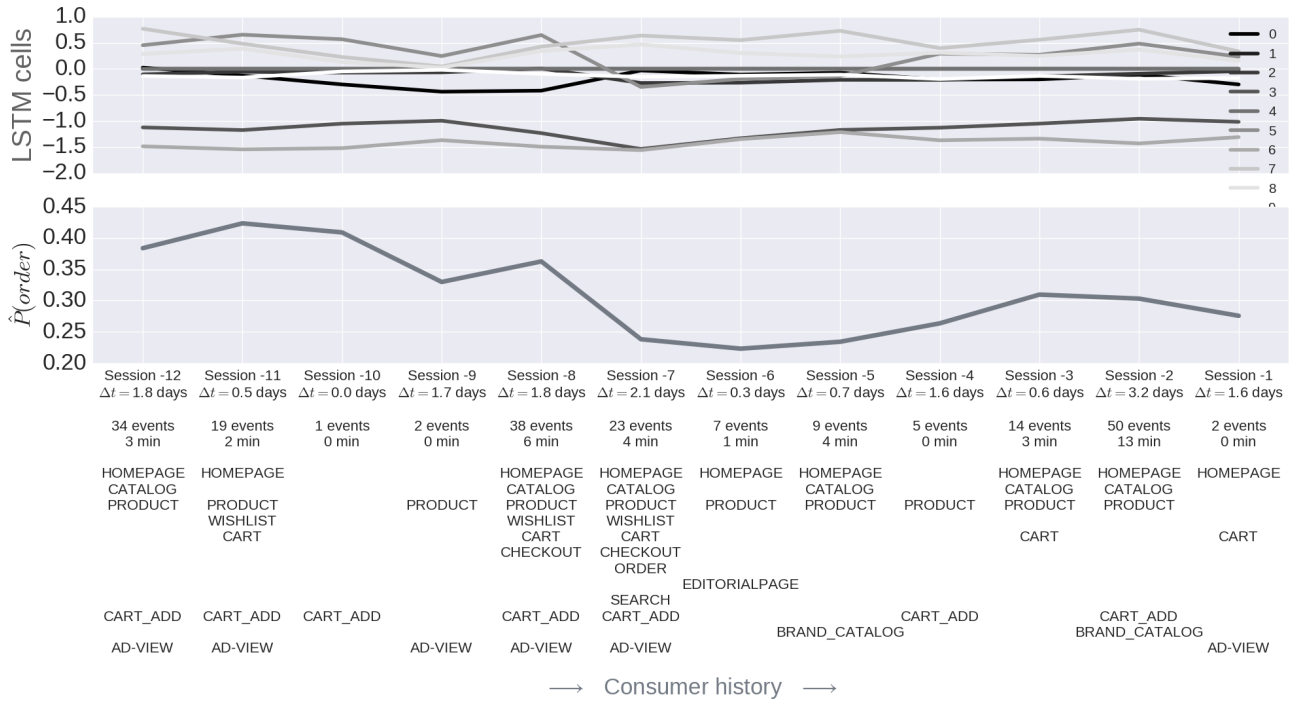

Figure 3: The session journey of a consumer is shown. We can explain the predictions of a session-stream RNN for the consumer by linking her sessions with LSTM cell states and predictions. The input to the RNN for each session consists of the time difference to the previous session; indicators which action types appeared in the session; the total number of actions; and the session length in seconds (shown in minutes here).

[10] M. Korpusik, S. Sakaki, F. Chen, and Y. Chen. Recurrent neural networks for customer purchase prediction on twitter. In *Workshop New Trends in Content-Based Recommender Systems*, 2016.

[11] V. Krakovna and F. Doshi-Velez. Increasing the interpretability of recurrent neural networks using hidden Markov models. In *Workshop on Interpretable Machine Learning in Complex Systems*, NIPS, 2016.

[12] Nicholas Leonard, Sagar Waghmare, Yang Wang, and Jin-Hwa Kim. rnn: Recurrent library for torch. 2015. arXiv preprint arXiv:1511.07889. Accessed on 6 Dec 2016.

[13] Z. Lipton. The mythos of model interpretability. In *ICML Workshop on Human Interpretability in Machine Learning*, 2016.

[14] M. Ribeiro, S. Singh, and C. Guestrin. "Why should I trust you?": Explaining the predictions of any classifier. In *Conf. on Knowledge Discovery and Data Mining (KDD)*, 2016.

[15] Y. Richter, E. Yom-Tov, and N. Slonim. Predicting customer churn in mobile networks through analysis of social groups. In *SIAM Intern. Conf. on Data Mining*, 2010.

[16] A. Vellido, J.D. Martin, F. Rossi, and P.J.G. Lisboa. Seeing is believing: The importance of visualization in real-world machine learning applications. In Verleysen, editor, *European Symposium on Artificial Neural Networks (ESANN)*, pages 219–226, 2011.

[17] A. Vieira. Predicting online user behaviour using deep learning algorithms. 2016. http://arxiv.org/abs/1511.06247. Accessed on 6 Dec 2016.

[18] A. Wangperawong, C. Brun, O. Laudy, and R. Pavasuthipaisit. Churn analysis using deep convolutional neural networks and autoencoders. 2016. https://arxiv.org/pdf/1604.05377.pdf. Accessed on 6 Dec 2016.

[19] Y. Zhang, H. Dai, C. Xu, J. Feng, T. Wang, J. Bian, B. Wang, and T. Liu. Sequential click prediction for sponsored search with recurrent neural networks. In *AAAI Conf. on Artificial Intelligence*, pages 1369–1375, 2014.