

Predicting purchasing intent: Automatic Feature Learning using Recurrent Neural Networks

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 a neural network for predicting purchasing intent in an Ecommerce setting. Our main contribution is to address the significant investment in feature engineering that is usually associated with state-of-the-art methods such as Gradient Boosted Machines. We use trainable vector spaces to model varied, semi-structured input data comprising categoricals, quantities and unique instances. Multi-layer recurrent neural networks capture both session-local and dataset-global event dependencies and relationships for user sessions of any length. An exploration of model design decisions including parameter sharing and skip connections further increase model accuracy. Results on benchmark datasets deliver classification accuracy within 98% of state-of-the-art on one and exceed state-of-the-art on the second without the need for any domain / dataset-specific feature engineering on both short and long event sequences.

KEYWORDS

Ecommerce, Deep Learning, Recurrent Neural Networks, Long Short Term Memory (LSTM), Embedding, Vector Space Models

ACM Reference Format:

Humphrey Sheil, Omer Rana, and Ronan Reilly. 2018. Predicting purchasing intent: Automatic Feature Learning using Recurrent Neural Networks. In *Proceedings of ACM SIGIR Workshop on eCommerce (SIGIR 2018 eCom)*. ACM, New York, NY, USA, 9 pages. <https://doi.org/10.1145/nnnnnnn.nnnnnnn>

1 INTRODUCTION

In the Ecommerce domain, merchants can increase their sales volume and profit margin by acquiring better answers for two questions:

- Which users are most likely to purchase (predict purchasing intent).
- Which elements of the product catalogue do users prefer (rank content).

By how much can merchants realistically increase profits? Table 1 illustrates that merchants can improve profit by between 2% and 11% depending on the contributing variable. In the fluid and highly competitive world of online retailing, these margins are significant, and understanding a user's shopping intent can positively

influence three out of four major variables that affect profit. In addition, merchants increasingly rely on (and pay advertising to) much larger third-party portals (for example eBay, Google, Bing, Taobao, Amazon) to achieve their distribution, so any direct measures the merchant group can use to increase their profit is sorely needed.

	McKinsey	A.T. Kearney	Affected by shopping intent
Price management	11.1%	8.2%	Yes
Variable cost	7.8%	5.1%	Yes
Sales volume	3.3%	3.0%	Yes
Fixed cost	2.3%	2.0%	No

Table 1: Effect of improving different variables on operating profit, from [22]. In three out of four categories, knowing more about a user's shopping intent can be used to improve merchant profit.

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 grouping of these tuples by user ID is commonly known as a *session*.

Predicting a users intent to purchase is more difficult than ranking content for the following reasons [29]: 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. In our opinion, this user behaviour is in stark contrast to other settings such as predicting if a user will "like" or "pin" a piece of content hosted on a social media platform after viewing it, where there is no monetary amount at stake for the user. As noted in [31], 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 proactive actions to maximize 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.

However there are counterweights to the desire to create more and more accurate models of online user behaviour - namely user privacy and ease of implementation. Users are increasingly reluctant to share personal information with online services, while complex Machine Learning models are difficult to implement and maintain in a production setting [28].

We surveyed existing work in this area and found that well-performing approaches have a number of factors in common:

- Heavy investment in dataset-specific feature engineering was necessary, regardless of the model implementation chosen.
- Model choices favour techniques such as Gradient Boosted Machines (GBM) [7] and Field-aware Factorization Machines (FFM) [24] which are well-suited to creating representations of semi-structured clickstream data once good features have been developed [26], [36], [34].

In [29], an important feature class employed the notion of item similarity, modelled as a learned vector space generated by word2vec [18] and calculated using a standard pairwise cosine metric between item vectors. In an Ecommerce context, items are more similar if they co-occur frequently over all user sessions in the corpus and are dissimilar if they infrequently co-occur. The items themselves may be physically dissimilar (for example - headphones and batteries), but they are often browsed and purchased together.

However, in common with other work, [29] still requires a heavy investment in feature engineering. The drawback of specific features is how tied they are to either a domain, dataset or both. The ability of Deep Learning to discover good representations without explicit feature engineering is well-known [8]. In addition, Artificial neural networks (ANNs) perform well with distributed representations such as embeddings, and ANNs with a recurrence capability to model events over time - Recurrent Neural Networks (RNNs) - are well-suited to sequence processing and labelling tasks [16].

Our motivation then is to build a good model of user intent prediction which does not rely on private user data, and is also straightforward to implement in a real-world environment. What performance can RNNs with an appropriate input representation and end-to-end training regime achieve on the prediction of purchasing intent task? Can this performance be achieved within the constraint of only processing anonymous session data and remaining straightforward to implement on other Ecommerce datasets?

2 RELATED WORK

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. [26] was the original winner of the competition using one of the datasets considered here using a commercial implementation of GBM with extensive feature engineering and is still to our knowledge the State of the Art (SOTA) implementation for this dataset. However the paper authors did make their model predictions freely available and we use these in the Experiments section to compare our model performance to theirs.

[10] 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. [17] compares [10] to a variety of classical Machine Learning algorithms on multiple datasets and finds that performance varies considerably by dataset. [37] extends [10] 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 multiple ways. For shopping behaviour prediction, [31] 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 [12], 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. [23] adopts a tangential approach - still focused on predicting purchases, but using textual product metadata to correlate words and terms that suit a particular geographic market better than others. Broadening our focus to include the general use of RNNs in the Ecommerce domain, Recurrent Recommender Networks are used in [35] to incorporate temporal features with user preferences to improve recommendations, to predict future behavioural directions, but not purchase intent. [30] further extends [10] by focusing on data augmentation and compensating for shifts in the underlying distribution of the data.

In [15], the authors augment a more classical Machine Learning approach (Singular Value Decomposition or SVD) to better capture temporal information to predict user behaviour - an alternative approach to the unrolling methodology used in this paper.

Using embeddings as a learned representation is a common technique. In [1], embeddings are used to model items in a low dimensional space to calculate a similarity metric, however temporal ordering is discarded. Learnable embeddings are also used in [9] to model items and used purchase confirmation emails as a high-quality signal of user intent. Unrolling events that exceed an arbitrary threshold to create a better input representation for user dwelltime or interest is addressed in [3]. In [6], Convolutional Neural Networks (CNNs) are used as the model implementation and micro-blogging content is analyzed rather than an Ecommerce clickstream.

3 OUR APPROACH

Classical Machine Learning approaches such as GBM work well and are widely used on Ecommerce data, at least in part because the data is *structured*. GBM is an efficient model as it enables an additive expansion in a set of basis functions or weak learners to continually minimize a residual error. One weakness of GBM is a propensity for overly-deep or wide decision trees to over-fit the training data and thus record poor performance on the validation and test set due to high variance [33], [34]. although this can be controlled using hyperparameters (namely tree depth, learning rate, minimum weight to split a tree node (`min_child_weight`) and data sub-sampling). GBM also requires significant feature engineering effort and does not naturally process the sequence in order, rather it consumes a compressed version of it (although it is possible to

provide a one-hot vector representation of the input sequence as a feature). Our approach is dual in nature - firstly we construct an input representation for clickstream / session data that eliminates the need for feature engineering. Second, we design a model which can consume this input representation and predict user purchase intent in an end-to-end, sequence to prediction manner.

3.1 Embeddings as item / word representations

Natural Language Processing (NLP) tasks, such as information retrieval, part-of-speech tagging and chunking, operate by assigning a probability value to a sequence of words. To this end, language models have been developed, defining a mathematical model to capture statistical properties of words and the dependencies among them.

Learning good representations of input data is a central task in designing a machine learning model that can perform well. An *embedding* is a vector space model where words are converted to a low-dimensional vector. Vector space models embed words where semantically similar words are mapped to nearby points. Popular generators of word to vector mappings such as [18] and [21], operate in an unsupervised manner - predicting similarity or minimizing a perplexity metric using word co-occurrence counts over a target corpus. We decided to employ embeddings as our target representation since:

- We can train the embeddings layer at the same time as training the model itself - promoting simplicity.
- Ecommerce data is straightforward to model as a dictionary of words.
- Embedding size can be increased or decreased based on dictionary size and word complexity during the architecture tuning / hyper parameter search phase.

Unlike [18] and [21], we chose not to pre-train the embeddings to minimize a perplexity error measure. Instead we allow the model to modify the embedding weights at training time by back-propagating the loss from a binary classification criterion.

3.2 Recurrent Neural Networks

Recurrent neural networks [27] (RNNs) are a specialized 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 $t = 1 \dots T$ is thought of as time and $l = 1 \dots L$ is the depth. All intermediate vectors h_t^l are computed as a function of h_{t-1}^l and h_{t-1}^{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, \dots, x_t [13].

3.2.1 LSTM and GRU. Long Short-Term Memory (LSTM) [11] is an extension to colloquial or vanilla RNNs designed to address the twin problems of vanishing and exploding gradients during training [19]. 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. [8] explain this

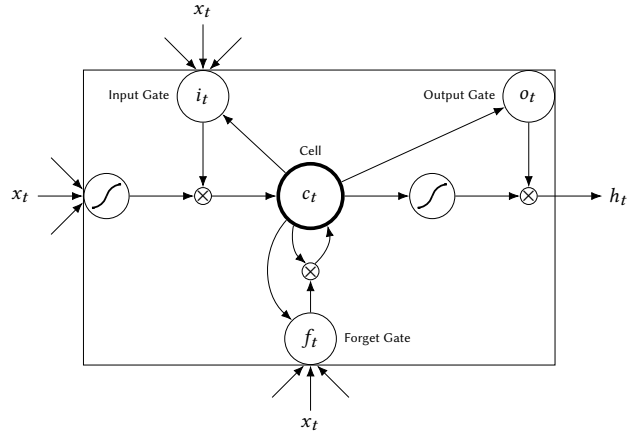


Figure 1: A single LSTM cell, depicting the hidden and cell states, as well as the three gates controlling memory (input, forget and output).

phenomenon succinctly. Like all deep learning models, RNNs require multiplication by a matrix W . After t steps, this equates to multiplying by W^t . Therefore:

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

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 $\text{diag}(\lambda)^t$.

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

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

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.

Gated Recurrent Units, or GRU [5] are a simplification of LSTM, with one less gate and the hidden state and cell state vectors combined. In practice, both LSTM and GRU are used interchangeably and the performance difference between both cell types is often minimal and / or dataset-specific.

4 IMPLEMENTATION

4.1 Datasets used

The RecSys 2015 Challenge [2] and the Retail Rocket Kaggle [25] datasets provide anonymous Ecommerce clickstream data well suited to testing purchase prediction models. Both datasets are reasonable in size - consisting of 9.2 million and 1.4 million user sessions respectively. 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 logic used to mark the start and end of a user session is dataset-specific - the RecSys 2015 dataset contains more sessions with a small item catalogue while the Retail Rocket dataset contains less sessions with an item catalogue 5x larger than the RecSys 2015 dataset. Both datasets contain a very high proportion of short length sessions (≤ 3 events), making this problem setting quite difficult for RNNs to solve. The Retail Rocket dataset contains much longer sessions when measured by duration - the RecSys 2015 user sessions are much shorter in duration. In summary, the datasets differ in important respects, and provide a good test of model generalization ability.

For both datasets, no sessions were excluded - both datasets in their entirety were used in training and evaluation. 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 much harder for our RNN model, but is a fairer comparison to previous work using GBM where all session lengths were also included [26],[34],[29].

The RecSys 2015 Challenge dataset includes a dedicated test set, while the Retail Rocket dataset does not. We reserved 20% of the Retail Rocket dataset for use in prediction / evaluation - the same proportion as the RecSys 2015 Challenge test dataset.

	RecSys 2015	Retail Rocket
Sessions	9,249,729	1,398,795
Buyer sessions	5.5%	0.7%
Unique items	52,739	227,006

Table 2: A short comparison of the two datasets used - RecSys 2015 and Retail Rocket.

4.2 Data preparation

The RecSys 2015 challenge dataset consists of 9.2 million user-item click sessions. Sessions are anonymous and classes are imbalanced with only 5% of sessions ending in one or more buy events. Each user session captures the interactions between a single user and items or products : $S_n = e_1, e_2, \dots, e_k$, where e_k is either a click or buy event. An example 2-event session is:

Both datasets contain missing or obfuscated data - presumably for commercially sensitive reasons. Where sessions end with one or more purchase events, the item price and quantity values are provided only 30% of the time in the case of the RecSys 2015 dataset, while prices are obfuscated for commercial reasons in the Retail

SID	Timestamp	Item ID	Cat ID
1	2014-04-07T10:51:09.277Z	214536502	0
1	2014-04-07T10:57:09.868Z	214536500	0

Table 3: An example of a clicker session from the RecSys 2015 dataset.

Rocket dataset. Therefore these elements of the data provide limited value.

SID	Item ID	Price	Quantity
420374	214537888	12462	1
420374	214537850	10471	1

Table 4: An example of the buy events from a buyer session (timestamp column elided for brevity).

The Retail Rocket dataset consists of 1.4 million sessions. Sessions are also anonymous and are even more imbalanced - just 0.7% of the sessions end in a buy event. This dataset also provides item metadata but in order to standardize our approach across both datasets, we chose not to use any information that was not common to both datasets. In particular we discard and do not use the additional "addtobasket" event that is present in the Retail Rocket dataset. Since it is so closely correlated with the buy event (users add to a basket before purchasing that basket), it renders the buyer prediction task trivial and an AUC of 0.97 is easily achievable for both our RNN and GBM models.

Our approach in preparing the data for training is as follows. We process each column as follows:

- Session IDs are discarded (of course we retain the sequence grouping indicated by the IDs).
- Timestamps are quantized into bins 4 hours in duration.
- Item IDs are unchanged.
- Category IDs are unchanged.
- Purchase prices are unchanged. We calculate price variance per item to convey price movements to our model (e.g. a merchant special offer).
- Purchase quantities are unchanged.

Each field is then converted to an embedding vocabulary - simply a lookup table mapping values to integer IDs. We do not impose a minimum occurrence limit on any field - a value occurring even once will be represented in the respective embedding. This ensures that even "long tail" items will be presented to the model during training. Lookup tables are then converted to an embedding with embedding weights initialized from a range $\{-0.075, +0.075\}$ - Table 5 identifies the number of unique items per embedding and the width used. The testing dataset contains both item IDs and category IDs that are not present in the training set - however only a very small number of sessions are affected by this data in the test set.

This approach, combined with the use of Artificial Neural Networks, provides a learnable capacity to encode more information than just the original numeric value. For example, an item price of \$100 vs \$150 is not simply a numeric price difference, it can also signify learnable information on brand, premium vs value and so on.

Data name	Train	Train+Test	Embedding Width
Item ID	52,739	54,287	100
Category ID	340	348	10
Timestamp	4,368	4,368	10
Price	667	667	10
Quantity	1	1	10

Table 5: Data field embeddings and dimensions, along with unique value counts for the training and test splits of the RecSys 2015 dataset.

Dataset	Events before	Events after	% increase
RecSys 2015	41,255,735	56,059,913	36%
Retail Rocket	2,351,354	11,224,267	377%

Table 6: Effect of unrolling by dwelltime on the RecSys 2015 and Retail Rocket datasets. There is a clear difference in the mean / median session duration of each dataset.

4.3 Event Unrolling

In [3], a more explicit representation of user dwelltime or interest in a particular item i_k in a sequence e_{i_1}, \dots, e_{i_k} is provided to the model by repeating the presentation of the event containing the item to the model in proportion to the elapsed time between e_{i_k} and $e_{i_{k+1}}$. In the example 2-event session displayed previously, the elapsed time between the first and second event is 6 minutes, therefore the first event is replayed 3 times during training and inference ($\lceil 360/150 \rceil$). In contrast to [3], we found that session unrolling provided a smaller improvement in model performance - for example on the RecSys 2015 dataset our best AUC increased from 0.837 to 0.839 when using the optimal unrolling value (which we discovered empirically using grid search) of 150 seconds. Unrolling also comes with a cost of increasing session length and thus training time - Table 6 demonstrates the effect of session unrolling on the size of the training / validation and test sets on both datasets.

4.4 Sequence Reversal

From [29], we know that the most important item in a user session is the last item (followed by the first item). Intuitively, the last and first items a user browses in a session are most predictive of purchase intent. To capitalize on this, we reversed the sequence order for each session before presenting them as batches to the model. Re-ordering the sequences provided an increase in test AUC on the RecSys 2015 dataset of 0.005 - from 0.834 to 0.839.

4.5 Model

4.5.1 Model Architecture. The data embedding modules are concatenated and presented to a configurable number of RNN layers (typically 3), with a final linear layer combining the output of the hidden units from the last layer. A sigmoid function is then applied to calculate a confidence probability in class membership. The model is trained by minimizing an unweighted binary cross entropy loss:

$$l_n = -[y_n \cdot \log x_n + (1 - y_n) \cdot \log(1 - x_n)] \quad (3)$$

where:

x_n is the output label value from the model $[0..1]$

y_n is the target label value $\{0, 1\}$.

We conducted a grid search over the number of layers and layer size by RNN type, as indicated in Table 7 below.

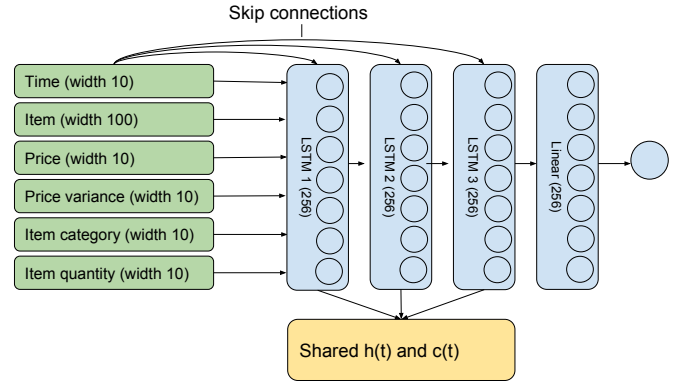


Figure 2: Model architecture used - the output is interpreted as the log probability that the input contains either a clicker or buyer session. Skip connections are used to combine the original input with successive layer outputs, and each layer shares the same hidden layer parameters.

4.5.2 Hidden layer parameter sharing. One model design decision worthy of elaboration is how hidden information (and cell state for LSTM) is shared between the RNN layers. We found that best results were obtained by re-using hidden state across RNN layers - i.e. re-using the hidden state vector (and cell state vector for LSTM) from the previous layer in the architecture and initializing the next layer with these state vectors before presenting the next layer with the output from the previous layer. Our intuition here is that for Ecommerce datasets, the features learned by each layer are closely related due to the fact that Ecommerce log / clickstream data is very structured, therefore re-using hidden and cell states from lower layers helps higher layers to converge on important abstractions. Aggressive sharing of the hidden layers led to a very significant improvement in AUC, increasing from 0.75 to 0.84 in our single best model.

5 EXPERIMENTS AND RESULTS

In this section we describe the experimental setup, and the results obtained when comparing our best RNN model to GBM State of the Art, and a comparison of different RNN variants (vanilla, GRU, LSTM).

5.1 Training Details

Both datasets were split into a training set and validation set in a 90:10 ratio. The model was trained using the Adam optimizer [14], coupled with a binary cross entropy loss metric and a learning rate annealer. Training was halted after 2 successive epochs of worsening validation AUC. Table 8 illustrates the main hyperparameters and setting used during training.

Layers	RNN			GRU			LSTM		
	1	2	3	1	2	3	1	2	3
Layer size									
64	0.72	0.81	0.81	0.741	0.832	0.833	0.735	0.832	0.831
128	0.72	0.80	0.80	0.755	0.833	0.833	0.729	0.834	0.834
256	0.71	0.80	0.80	0.732	0.834	0.834	0.724	0.834	0.839
512	0.69	0.80	0.77	0.746	0.832	0.833	0.759	0.835	0.839

Table 7: Model grid search results for number and size of RNN layers by RNN type on the RecSys 2015 dataset. The State of the Art baseline for comparison is 0.853.

Dataset split	90/10 (training / validation)
Hidden units range	128 – 512 : 256 optimal
Embedding width	10 – 300 : 100 optimal for items
Embedding weight	–0.075 to +0.075
Batch size	32 – 256 : 256 optimal for speed and regularization
Optimizer	Adam, learning rate $1e^{-3}$

Table 8: Hyper parameters and setup employed during model training.

We tested three main types of recurrent cells (vanilla RNN, GRU, LSTM) as well as varying the number of cells per layer and layers per model. While a 3-layer LSTM achieved the best performance, vanilla RNNs which possess no memory mechanism are able to achieve a competitive AUC. The datasets are heavily weighted towards shorter session lengths (even after unrolling - see Figure 4 and 9). We posit that the power of LSTM and GRU is not needed for the shorter sequences of the dataset, and colloquial recurrence with embeddings has the capacity to model sessions over a short enough sequence length.

5.2 Overall results

The metric we used in our analysis was Area Under the ROC Curve or AUC. AUC is insensitive to class imbalance, and also the raw predictions from [26] were available to us, thus a detailed, like-for-like AUC comparison using the test set is the best model comparison. The organizers of the challenge also released the solution to the challenge, enabling the test set to be used. After training, the LSTM model AUC obtained on the test set was 0.839 - 98.4% of the AUC (0.853) obtained by the SOTA model. As the subsequent experiments demonstrate, a combination of feature embeddings and model architecture decisions contribute to this performance. For all model architecture variants tested (see Table 7), the best performance was achieved after training for a small number of epochs (2 - 3). This held true for both datasets.

Our LSTM model achieved within 98% of the SOTA GBM performance on the RecSys 2015 dataset, and outperformed our GBM model by 0.5% on the Retail Rocket dataset, as table 9 shows.

5.3 Analysis

We constructed a number of tests to analyze model performance based on interesting subsets of the test data.

	RecSys 2015	Retail Rocket
LSTM	0.839	0.838
GBM	0.853	0.834

Table 9: Classification performance measured using Area under the ROC curve (AUC) of the GBM and LSTM models on the RecSys 2015 and Retail Rocket datasets.

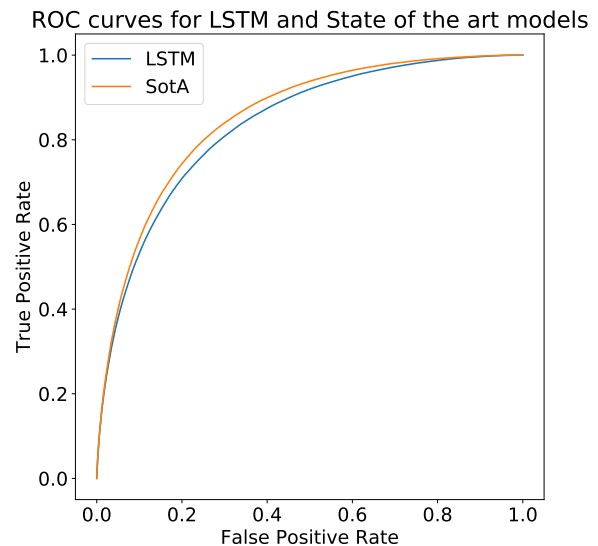


Figure 3: ROC curves for the LSTM and State of the Art models - on the RecSys 2015 test set.

5.3.1 Session length. Figure 4 graphs the best RNN model (a 3-layer LSTM with 256 cells per layer) and the SOTA model, with AUC scores broken down by session length. For context, the quantities for each session length in the test set is also provided. Both models underperform for sessions with just one click - clearly it is difficult to split clickers from buyers with such a small input signal. For the remaining session lengths, the relative model performance is consistent, although the LSTM model does start to close the gap after sessions with length > 10.

5.3.2 User dwelltime. Given that we unrolled long-running events in order to provide more input to the RNN models, we evaluated

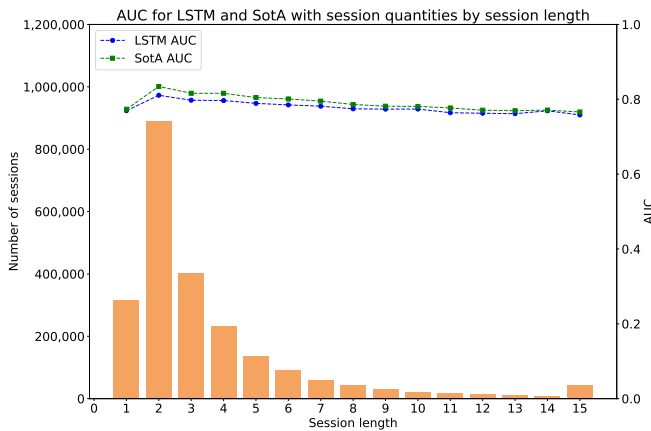


Figure 4: AUC by session length for the LSTM and SOTA models, session quantities by length also provided for context - clearly showing the bias towards short sequence / session lengths in the RecSys 2015 dataset.

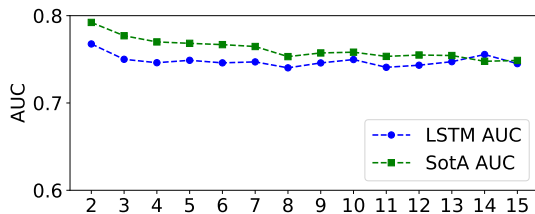


Figure 5: AUC by session length for the LSTM and SOTA models, for any sessions where unrolling by dwelltime was employed. There are no sessions of length 1 as unrolling is inapplicable for these sessions (RecSys 2015 dataset).

the relative performance of each model when presented with sessions with any dwelltime > 1. As Figure 5 shows, LSTM is closer to SOTA for this subset of sessions and indeed outperforms SOTA for session length = 14, but the volume of sessions affected (5,433) is not enough to materially impact the overall AUC.

5.3.3 *Item price.* Like most Ecommerce catalogues, the catalogue under consideration here displays a considerable range of item prices. We first selected all sessions where any single item price was > 10,000 (capturing 544,014 sessions) and then user sessions where the price was <= 750 (roughly 25% of the maximum price - capturing 1,063,034 sessions). Figures 6 and 7 show the relative model performance for each session grouping. As with other selections, the relative model performance is broadly consistent - there is no region where LSTM either dramatically outperforms or underperforms the SOTA model.

5.3.4 *Gated vs un-gated RNNs.* Table 7 shows that while gated RNNs clearly outperform un-gated or vanilla RNNs, the difference is 0.02 of AUC which is less than might be expected. We believe the reason for this is that the dataset contains many more shorter

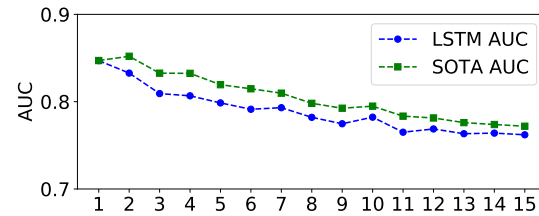


Figure 6: Model performance for sessions containing low price items, split by session length (RecSys 2015 dataset).

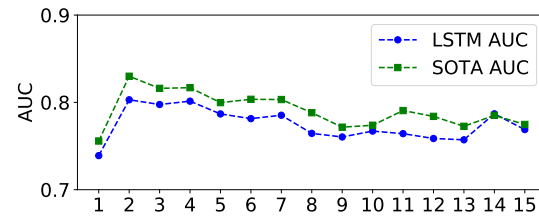


Figure 7: Model performance for sessions containing high price items, split by session length (RecSys 2015 dataset).

(< 5) than longer sequences. This is to be expected for anonymized data - user actions are only aggregated based on a current session token and there is no "lifetime" set of user events. For many real-world cases then, using un-gated RNNs may deliver acceptable performance.

5.3.5 *End-to-end learning.* To measure the effect of allowing (or not) the gradients from the output loss to flow unencumbered throughout the model (including the embeddings), we froze the embedding layer so no gradient updates were applied and then trained the network. Model performance decreased to an AUC of 0.808 and training time increased by 3x to reach this AUC. Depriving the model of the ability to dynamically modify the input data representation using gradients derived from the output loss metric reduces its ability to solve the classification problem posed.

5.4 Transferring to another dataset

The GBM model itself used in [26] is not publically available, however we were able to use the GBM model described in [29]. Figure 8 shows the respective ROC curves for the RNN (LSTM) and GBM models when they are ported to the Retail Rocket dataset. Both models still perform well, however the LSTM model slightly outperforms the GBM model (AUC of 0.837 vs 0.834).

A deeper analysis of the Area under the ROC curve demonstrates how the characteristics of the dataset can impact on model performance. The Retail Rocket dataset is heavily weighted towards single-click sessions as Figure 9 shows. LSTM outperforms GBM for these sessions - which can be attributed more to the learned embeddings since there is no sequence to process. GBM by contrast can extract only limited value from single-click sessions as important feature components such as dwelltime, similarity etc. are unavailable.

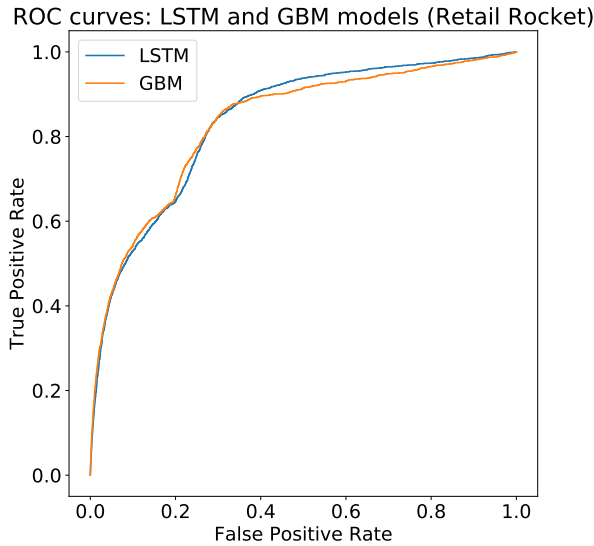


Figure 8: Area under ROC curves for LSTM and GBM models when ported to the Retail Rocket dataset. On this dataset, the LSTM model slightly outperforms the GBM model overall.

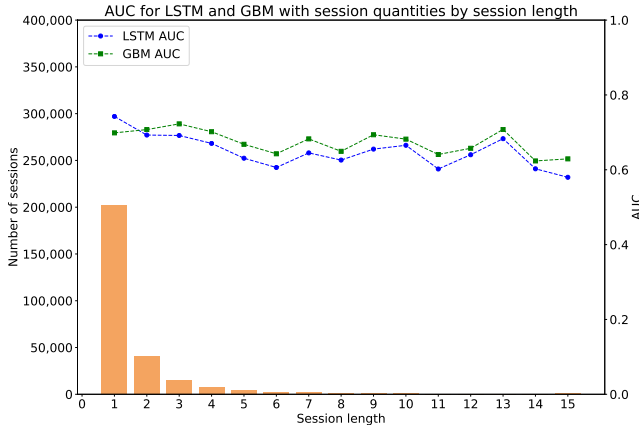


Figure 9: AUC by session length for the LSTM and GBM models when tested on the Retail Rocket dataset. The bias towards shorter sessions is even more prevalent versus the RecSys 2015 dataset.

5.5 Training time and resources used

We used PyTorch [20] to construct and train the LSTM models while XGBoost [4] was used to train the GBM models. The LSTM implementation was trained on a single Nvidia GeForce GTX TITAN Black (circa 2014 and with single-precision performance of 5.1 TFLOPs vs a 2017 GTX 1080 Ti with 11.3 TFLOPs) and consumed

between 1 and 2 GB RAM. The results reported were obtained after 3 epochs of training on the full dataset - an average of 6 hours (2 hours per epoch). This compares favourably to the training times and resources reported in [26], where 150 machines were used for 12 hours. However, 12 hours denotes the time needed to train two models (session purchase and item ranking) whereas we train just one model. While we cannot compare GBM (which was trained on a CPU cluster) to RNN (trained on a single GPU with a different parallelization strategy) directly, we note that the hardware resources required for our model are modest and hence accessible to almost any commercial or academic setup. In addition, real-world Ecommerce datasets are large [32] and change rapidly, therefore usable models must be able to consume large datasets and be re-trained readily to cater for new items / documents.

5.6 Conclusions and future work

We presented a Recurrent Neural Network (RNN) model which recovers 98.4% of current SOTA performance on the user purchase prediction problem in Ecommerce without using explicit features. On a second dataset, our model fractionally exceeds SOTA performance. The model is straightforward to implement, generalizes to different datasets with comparable performance and can be trained with modest hardware resource requirements.

It is promising that gated RNNs with no feature engineering can be competitive with Gradient Boosted Machines on short session lengths and structured data - GBM is a more established model choice in the domain of Recommender Systems and Ecommerce in general. We believe additional work on input representation (while still avoiding feature engineering) can further improve results for both gated and non-gated RNNs. One area of focus will be to investigate how parameter sharing at the hidden layer helps RNNs to operate on short sequences of structured data prevalent in Ecommerce.

Lastly, we note that although our approach requires no feature engineering, it is also inherently transductive - we plan to investigate embedding generation and maintenance approaches for new unseen items / documents to add an inductive capability to the architecture.

6 ACKNOWLEDGEMENTS

We would like to thank the authors of [26] for making their original test submission available and the organizers of the original challenge in releasing the solution file after the competition ended, enabling us to carry out our comparisons.

REFERENCES

- [1] Oren Barkan and Noam Koenigstein. 2016. Item2Vec: Neural Item Embedding for Collaborative Filtering. *CoRR* abs/1603.04259 (2016). arXiv:1603.04259 <http://arxiv.org/abs/1603.04259>
- [2] David Ben-Shimon, Alexander Tsikinovsky, Michael Friedmann, Bracha Shapira, Lior Rokach, and Johannes Hoerle. 2015. RecSys Challenge 2015 and the YOO-CHOOSE 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>
- [3] Veronika Bogina and Tsvi Kuflik. 2017. Incorporating Dwell Time in Session-Based Recommendations with Recurrent Neural Networks. In *RecTemp@RecSys*.
- [4] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. ACM, New York, NY, USA, 785–794. <https://doi.org/10.1145/2939672.2939785>

- [5] KyungHyun Cho, Bart van Merriënboer, 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>
- [6] Xiao Ding, Ting Liu, Junwen Duan, and Jian-Yun Nie. 2015. Mining User Consumption Intention from Social Media Using Domain Adaptive Convolutional Neural Network. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI'15)*. AAAI Press, 2389–2395. <http://dl.acm.org/citation.cfm?id=2886521.2886653>
- [7] Jerome H. Friedman. 2000. Greedy Function Approximation: A Gradient Boosting Machine. *Annals of Statistics* 29 (2000), 1189–1232.
- [8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. <http://www.deeplearningbook.org>.
- [9] Mihajlo Grbovic, Vladan Radosavljevic, Nemanja Djuric, Narayan Bhamidipati, Jaikrit Savla, Varun Bhagwan, and Doug Sharp. 2016. E-commerce in Your Inbox: Product Recommendations at Scale. *CoRR* abs/1606.07154 (2016). arXiv:1606.07154 <http://arxiv.org/abs/1606.07154>
- [10] 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>
- [11] 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>
- [12] 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.
- [13] 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>
- [14] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014). arXiv:1412.6980 <http://arxiv.org/abs/1412.6980>
- [15] Yehuda Koren. 2009. Collaborative Filtering with Temporal Dynamics. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '09)*. ACM, New York, NY, USA, 447–456. <https://doi.org/10.1145/1557019.1557072>
- [16] 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>
- [17] 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>
- [18] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *CoRR* abs/1301.3781 (2013). arXiv:1301.3781 <http://arxiv.org/abs/1301.3781>
- [19] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the Difficulty of Training Recurrent Neural Networks. In *Proceedings of the 30th International Conference on Machine Learning - Volume 28 (ICML'13)*. JMLR.org, III–1310–III–1318. <http://dl.acm.org/citation.cfm?id=3042817.3043083>
- [20] 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).
- [21] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. 1532–1543. <http://www.aclweb.org/anthology/D14-1162>
- [22] R.L. Phillips. 2005. *Pricing and Revenue Optimization*. Stanford University Press. <https://books.google.co.uk/books?id=bXsyO06qikEC>
- [23] Reid Pryzant, Young joo Chung, and Dan Jurafsky. 2017. Predicting Sales from the Language of Product Descriptions. In *ACM SIGIR Forum*. ACM.
- [24] Steffen Rendle. 2010. Factorization Machines. In *Proceedings of the 2010 IEEE International Conference on Data Mining (ICDM '10)*. IEEE Computer Society, Washington, DC, USA, 995–1000. <https://doi.org/10.1109/ICDM.2010.127>
- [25] Retailrocket. 2017. Retailrocket recommender system dataset. <https://www.kaggle.com/retailrocket/ecommerce-dataset>. (2017). [Online; accessed 01-Feb-2018].
- [26] 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>
- [27] 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>
- [28] D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, and Michael Young. 2014. Machine Learning: The High Interest Credit Card of Technical Debt. In *SE4ML: Software Engineering for Machine Learning (NIPS 2014 Workshop)*.
- [29] 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
- [30] 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>
- [31] Arthur Toth, Louis Tan, Giuseppe Di Fabrizio, and Ankur Datta. 2017. Predicting Shopping Behavior with Mixture of RNNs. In *ACM SIGIR Forum*. ACM.
- [32] Andrew Trotman, Jon Degenhardt, and Surya Kallumadi. 2017. The Architecture of eBay Search. In *ACM SIGIR Forum*. ACM.
- [33] Alexander Vezhnevets and Olga Barinova. 2007. Avoiding Boosting Overfitting by Removing Confusing Samples. In *Proceedings of the 18th European Conference on Machine Learning (ECML '07)*. Springer-Verlag, Berlin, Heidelberg, 430–441. https://doi.org/10.1007/978-3-540-74958-5_40
- [34] 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>
- [35] 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>
- [36] Peng Yan, Xiaocong Zhou, and Yitao Duan. 2015. E-Commerce Item Recommendation Based on Field-aware Factorization Machine. In *Proceedings of the 2015 International ACM Recommender Systems Challenge (RecSys '15 Challenge)*. ACM, New York, NY, USA, Article 2, 4 pages. <https://doi.org/10.1145/2813448.2813511>
- [37] 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*.