Location Aware Sequential Recommendation Using Hybrid Recurrent Neural Networks

submitted in partial fulfillment for the degree of master of science

Dimitris Michailidis
12325929

master information studies
data science
faculty of science
university of amsterdam

2019-06-26

	Internal Supervisor	**External Supervisor**
Title, Name	Dr Pengjie Ren	Dr Dmitri Jarnikov
Affiliation	UvA, ILPS	Naspers Group
Email	p.ren@uva.nl	dmitri.jarnikov@naspers.com

Location Aware Sequential Recommendation Using Hybrid Recurrent Neural Networks

Dimitris Michailidis
jimichailidis@gmail.com
University of Amsterdam
Amsterdam, The Netherlands

ABSTRACT

Recently, Recurrent Neural Networks (RNNs) have been employed for the Sequential Recommendation task, showing a significant improvement over traditional systems. Research has focused on predicting the probability of the next items to be clicked in a session, using either plain item IDs or commonly found features like images, text and time. In this paper, we propose two families of architectures that perform location-aware recommendations in the e-commerce domain: a parallel recurrent and a latent context embedding approach. We evaluate the architectures against the current state of the art methods and achieve improvement in terms of Recall and MRR.

KEYWORDS

Recommender Systems, Session-based Recommendation, Location Aware Recommendation, Click Prediction, Recurrent Neural Networks, Gated Recurrent Unit

1 INTRODUCTION

The rapid growth of online shopping has lead to the emergence of massive global marketplace platforms, offering hundreds of thousands of products, in constantly changing catalogues. Providing a smooth experience for users in such a dynamic setting is paramount for businesses who will to achieve a competitive edge. Recommender systems are being employed widely to achieve this goal. The idea is to study the previous actions of the customers, gain insights on their preferences and then display to them a personalized selection of items they are most likely to purchase next. Hence, understanding what drives user behavior is key for providing accurate recommendations. Fortunately, users expose their preferences through signals, either explicit (ratings, reviews), or implicit (impressions, clicks).

In recent years, research on recommendation have focused on models that leverage additional context provided by the sequences in which actions are being done by users. These methods are called Sequential Recommendation methods and have significantly improved traditional approaches. In Sequential Recommendation, interactions performed within the same time-frame are treated as a separate task, which is defined as follows. Given a sequence of user interactions in a session $[x_1, x_2, ..., x_{t-1}]$, find a model F that predicts the next most likely interaction $y = x_t = F([x_1, x_2, ..., x_{t-1}])$. It is common for x_i to be the current clicked item and y a ranking over all items in the dataset.

There are two basic challenges that such systems need to overcome: (1) the recommendations need to be accurate while relying on little information and (2) they need to operate in real time. Classical methods usually fail to meet those requirements, therefore researchers have shifted their focus on alternative approaches, with item-based ones being the most widely used [22, 33].

Recently, inspired by their performance on Natural Language Processing, researchers have successfully applied Recurrent Neural Network (RNN) architectures to the recommendation task, showing significant improvements over conventional methods [16, 35, 38]. The idea is to model each session in the same way as text, where each individual action is dependent on the previous ones and influences the next. Where e-commerce actions differ from plain text, however, is the additional context that affects the behavior of the user, such as product text, images, position on the screen, etc. The fact that simple RNNs ignore this context lead to a significant amount of follow up works focusing on incorporating it in the recommendation process, either in form of concatenation/fusion [5, 21, 36], or by modeling it independently in different networks [17, 25].

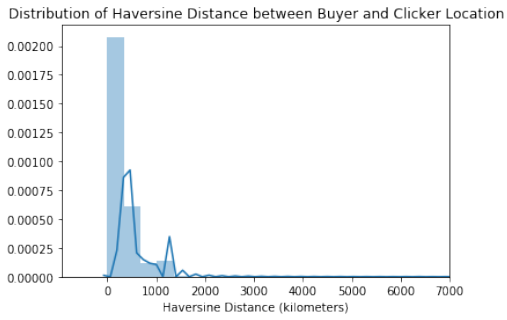

Figure 1: Users tend to click items that are in close proximity to their location. Data collected from OLX.

Most of the above studies focus either on visual features (text, images) or temporal dynamics (evolution, duration). In peer-to-peer marketplace platforms, customers pay attention to ease of access of items in terms of location, especially in cases when shipping is not always an option, e.g. when buying used cars. Figure 1 was constructed from interaction logs of visitors on a leading peer-to-peer marketplace. It shows that users tend to prefer clicking on products that are more accessible to them. In this study we propose architectures that aim to capture these dependencies and provide location-aware recommendations. Our main contributions are:

- We propose a parallel recurrent architecture that learns from both click behaviors and location information simultaneously.
- We propose a latent context architecture that enhances item embeddings with location context.

- We evaluate these architectures by performing experiments on an interaction dataset of a big e-commerce website.
- We analyse the effect of different location representation techniques and combination methods in the recommendation quality.

The rest of the paper is structured as follows: First, we discuss the focus of previous researches (Section 2). Then, we describe in detail the proposed architectures (Section 3) and the experimental setup used to evaluate them (Section 4). Finally, we present the results (Section 5), reflect on the findings (Section 6), and conclude with what should be explored next (Section 7).

2 RELATED WORK

In this section we describe previous research done on sequential (2.1), contextual (2.2) and location aware recommendation (2.3).

2.1 Sequential Recommendation

In the absence of reliable user history, early studies have approached the recommendation problem from the item-to-item perspective, where items that are purchased/clicked together in different sessions are deemed similar. Sarwar et al. [33] introduced several ways of creating similarity matrices and different models for obtaining recommendations. Linden et al. [22] further proved the scalability of the method by running offline and online experiments on Amazon. Desphande et al. [10] later on proposed a technique that uses conditional probabilities and is capable of differentiating between frequent and infrequent items. While effective, these methods ignore the sequential nature of the actions.

Traditional Approaches. Early work on sequential modeling explored the use of Markov chains, where each recommendation additionally considers previous actions through the use of states. Zidmars et al. [39] treat the problem as a univariate time series prediction by transforming the data and using decision-tree models to extract sequential patterns. Later on, Shani et al. [34] introduce a Markov Decision Process system that treats recommendations as actions and solves a first order Markov chain to recommend items using the transition probability between them. An approach that combines matrix factorization with Markov chains was introduced by Rendle et al. [32] where a transition matrix is learned separately for each session, and recommendation is done with a transition cube. Markov chains showed a great improvement over item-based methods, but modeling all possible sequences of user actions makes the state space impossible to manage.

Deep Learning Approaches. Recurrent Neural Networks (RNNs) have lately become increasingly popular, mainly due to their capabilities in modeling sequential dependencies in data. Distinct results have been achieved with the use of RNNs in machine translation [1, 2] and speech recognition [13, 20]. Inspired by their achievements in Natural Language Processing, researchers have applied the same concept on recommender systems.

Zhang et al. [38] found that user actions in a website largely depend on their previous behavior, by observing a positive correlation between last click dwell time and current click-through rate. They proposed a Recurrent Neural Network that learns to model the sequential dependency and predict which items are most likely to be clicked next by a user. They train the model using an averaged cross entropy loss and outperform logistic regression and feedforward network baselines. An alternative method was proposed by Hidasi et al. [16], where a Gated Recurrent Unit (GRU4Rec) is used and is trained with session parallel mini-batches. Moreover, they introduce the TOP1 pair-wise learning to rank loss function which is used widely in later researches as well. Follow up works focused on improving these models without dropping the performance and resource needs. Tan et al. [35] propose a data augmentation and distillation framework to effectively outperform GRU4Rec, while Jannach et al. [19] combine RNNs with neighborhood-based models. Recently, Hidasi et al. [15] introduced different negative sampling strategies that enhance ranking loss functions and outperform common baselines. Despite their significant contribution to improving the state of the art, these solutions mainly focus on click interactions, ignoring other context that influences how the users behave online.

2.2 Contextual Recommendation

Contextual recommendation aims to enhance recommender systems by employing additional human behavior traits; like focus of attention, preference evolution, device, location, etc. Most research has focused on how visual features and temporal dynamics affect recommendations. Bansal et al. [3] encode text sequences using GRUs for text based-recommendations, Quadrana et al. [17] introduce parallel architectures that learn from both text and image representations. Hidasi et al. [28] explore how preferences evolve in time with hierarchical architectures. Later on, Li et al. [21] introduce models that leverage duration of events to contextualize their embeddings, providing a more comprehensive input to the recurrent unit. Beutel et al. [5] propose an alternative approach, where time is fused in the model in different stages, including the hidden state of the recurrent neural network. Recent works in contextual recommendation include the attention-based approaches proposed by Zhang et al. [37] and Manotumruksa et al. [26], the model that aims to predict repeat consumption by Ren et al. [29] and the cross domain recommendation model by Ma et al. [25].

2.3 Location Aware Recommendation

A number of researches have been focusing on incorporating geospatial features in recommender systems. Starting from Point of Interest (POI) recommendation, Bao et al. [4] proposed a system that combines collaborative filtering with a heuristic spatial range candidate selector algorithm. They model location preferences from the visit history of both users and local experts and recommend a set of possible POIs to visit next. Cheng et al. [6] later introduce a sequence aware Markov chain model that captures both the temporal and the regional context simultaneously. Lu et al. [24] exploit GPS-tagged social media posts for the task of Expert Recommendation, by leveraging region, topic and social-based locality and integrating them into a matrix factorization model, outperforming popular baselines. In a follow up work, Ge et al. [11] apply the same philosophy in a tensor factorization model that learns from latent information based on topical and social preferences of users.

Although the above studies have been effective, they do not focus on learning from geospatial relationships between users in the sequential recommendation field. Contrastingly, this work aims

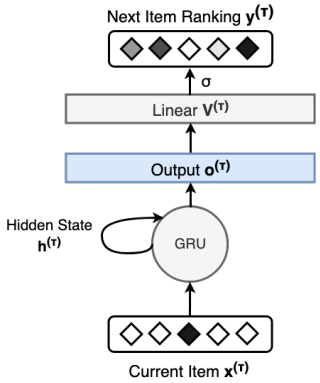

Figure 2: The typical GRU recommender architecture. The input is a 1-hot encoding of the current item and the output a ranking over all the items in the dataset. Adapted from [16].

to model these relationships and effectively provide more accessible recommendations to buyers.

3 METHODOLOGY

In this section we describe the proposed architectures. We begin by formalizing the problem mathematically (3.1), followed by the details of the baseline implementation (3.2). We then propose a location representation (3.3) and the two hybrid architectures (3.4, 3.5). Finally, we describe the details of the training process of the networks (3.6).

3.1 Preliminaries

As mentioned in Section 1, the task of a sequential recommender is to predict the next interaction (click, rating, purchase, etc.), given the previous history of interactions. These events can span from a single to multiple sequences in time. In this thesis we focus on session-based recommendation, therefore all actions correspond to a single session.

Let $s_i \in S$ be an individual, independent session of length t, consisting of a sequence of interactions $\{x_1, x_2, ..., x_t\}$. We consider x_t to be the interaction we want to predict and $x_{\bar{t}}$ the previous ones. The goal is therefore to approximate $y = f(x_{\bar{t}}) = x_t$, where f can be a matrix factorization model or a neural network. We further define y as predicting the following conditional probability:

$$P(x_t|x_{\bar{t}}) = P(x_t|\{x_1, x_2, ..., x_\tau, ..., x_{t-1}\}) \tag{1}$$

For the rest of the paper, $x_\tau \in \mathbb{N}\ (1 \leq \tau \leq N)$ is assumed to be a click action on a single item and is represented by the index of the current clicked item. We then convert this representation to an one-hot vector of size N, where all entries are 0 except for the one that corresponds to that index, which is 1. The output of the model, $f(x_{\bar{t}})$, is a ranking over all the items in the dataset.

3.2 Neural Recurrent Recommendation Baseline

Recurrent Neural Networks are the state of the art approaches for processing sequential data and are able to process variable length sequences [12]. They achieve that by sharing parameters across the different steps of the sequence, through the maintenance of an internal hidden unit. This hidden unit is commonly updated as follows:

$$h^{(\tau)} = g(b + Wh^{(\tau-1)} + Ux^{(\tau)}) \tag{2}$$

Where $h^{(\tau-1)}$ is the hidden state in the previous step, b is the bias, W and U are trainable weight matrices and g is a nonlinear activation function (typically sigmoid or hyperbolic tangent).

The hidden state $h^{(\tau)}$ is then used to calculate the output of the recurrent unit as follows:

$$o^{(\tau)} = b + Vh^{(\tau)} \tag{3}$$

Where V is also a trainable weight matrix. Finally, the output is passed through a sigmoid function to obtain the probability distribution over all the items in the dataset:

$$y^{(\tau)} = \sigma(o^{(\tau)}) \tag{4}$$

In practice, the most effective and commonly used RNNs are those based on the Gated Recurrent Unit (GRU) [8, 12], which were derived to combat the vanishing gradient problem of traditional RNNs [18]. A GRU differs from a conventional RNN in that it learns when and how to update its hidden state, using update and reset gates, defined in the following equations:

$$h^{(\tau)} = (1 - u^{(\tau)}) \cdot h^{(\tau-1)} + u^{(\tau)} \cdot c^{(\tau)} \tag{5}$$

$$u^{(\tau)} = \sigma(W_u x^{(\tau)} + U_u h^{(\tau-1)} + b) \tag{6}$$

$$c^{(\tau)} = tanh(W_c x^{(\tau)} + U_c(r^{(\tau)} \odot h^{(\tau-1)}) + b) \tag{7}$$

$$r^{(\tau)} = \sigma(W_r x^{(\tau)} + U_r h^{(\tau-1)} + b) \tag{8}$$

Where c is the candidate hidden state, u the update gate, r the reset gate, W, U trainable weight matrices and b biases. Figure 2 shows the typically used GRU-based recurrent neural architecture, as proposed by [16]. In this work we take this as a basis and extend it to incorporate geospatial relationships between buyers and sellers.

3.3 Representing Location

Before we train models that learn from the geospatial context, we need to convert this information into a suitable representation. Location can be represented using different types of data. The geographical coordinates of the buyer and the seller can be mapped to regional information like the country, the city and the district. This information can be modelled with a district representation (e.g. 1-hot vector) or an embedding. Distance can be modeled using the log-transformed haversine distance metric. A concatenated location representation can then be created as follows:

$$l = [a; c; d; d_{u,i}] \tag{9}$$

Where a, c, d are the 1-hot vectors of the country, city and district respectively and $d_{u,i} \in \mathbb{R}$ is the log distance between the user (u) and the item they interact with (i).

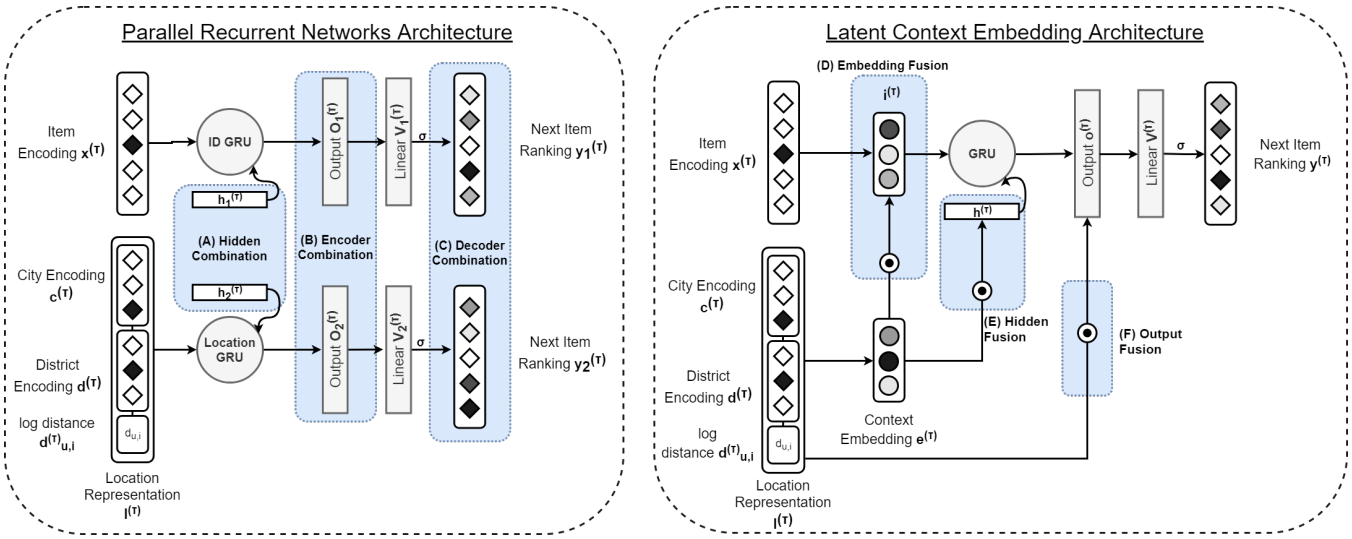

Figure 3: Proposed Hybrid Architectures. From left to right: (A) hidden state combination, (B) encoder combination, (C) decoder combination, (D) embedding fusion, (E) hidden fusion, (F) output fusion.

3.4 Parallel Recurrent Networks

We propose two families of hybrid architectures to learn from buyer and seller location relationships. The parallel architecture is built on the basis set in [17] and is designed to recurrently utilize both the item and the location representations. As shown in Figure 3 (left-side), it consists of two recurrent networks that train in parallel. One is the basic GRU introduced in [16] that learns directly from the clicked items and the other is a LocationGRU of the same configuration, which accepts as input the location representation. The networks communicate with each other in three different proposed stages.

(A) Hidden state combination. The hidden states of the two models are merged into a single one via the use of a weighted combination. The joint hidden state is then used to produce a single recurrent output. The hidden weights are learned independently, while a single joint weight matrix V is learned from both:

$$h^{(\tau)} = \alpha \cdot h_1 + (1 - \alpha) \cdot h_2 \tag{10}$$

$$o^{(\tau)} = V \cdot h^{(\tau)} \tag{11}$$

(B) Encoder combination. the recurrent units run independently and produce different outputs. These outputs are then combined and finally used as an input to the final activation layer, which produces a joint ranking.

$$o^{(\tau)} = \alpha \cdot o_1 + (1 - \alpha) \cdot o_2 \tag{12}$$

$$y^{(\tau)} = \sigma(o^{(\tau)}) \tag{13}$$

(C) Decoder Combination. both networks run independently and produce their own next-item ranking, which are then combined before the final activation, to create a joint ranking.

$$y_1^{(\tau)} = \sigma(o_1^{(\tau)}) \tag{14}$$

$$y_2^{(\tau)} = \sigma(o_2^{(\tau)}) \tag{15}$$

$$y^{(\tau)} = \alpha \cdot y_1 + (1 - \alpha) \cdot y_2 \tag{16}$$

3.5 Latent Context Embedding

The latent embedding approach is built upon the ideas by [5] and [21], in which the authors propose approaches to model the time context in sequential recommendation. As shown in Figure 3 (right-side), in contrast with the parallel approaches, location information here is not modelled recurrently. Instead, a latent context representation is created and then is fused with the base model, thus leading to a less complex model. Consequently, this approach tends to be more efficient than the parallel, because it only contains one recurrent network [5]. Similarly to 3.4, the fusion of the context representation is being done in three proposed stages.

(D) Embedding Fusion. The goal of this approach is to append the location information to each item click in the form of an embedding, thus creating a contextualized representation of the event, which captures both the clicked item and the location dependency at the same time. Recent work using this approach has shown improvements in neural machine translation [7] and sequential recommendation [21]. Firstly, a location context vector is computed, using the location representation (time steps are omitted for simplicity):

$$c_l = \phi(b + W_l \cdot l) \tag{17}$$

where ϕ is a non-linear activation and $z^{(\tau)}$ the location information of the current event. The context vector is then used as an input to a feedforward neural network, to create the location embedding.

$$e = \sigma(b + W_e \cdot c_l) \tag{18}$$

The location and item embeddings are then combined using an element-wise multiplication, which is then used as an input to the recurrent layer.

$$x_l = e \odot x \tag{19}$$

(E) Hidden Fusion. Here the location information is used to provide context to the hidden state of the recurrent network, instead of the input. It is based on the latent-cross proposed by [5] which improved Youtube's recurrent recommender. Instead of creating

a context embedding, location is used to learn a single-weighted representation, which is then multiplied with the recurrent hidden state before producing the output.

$$h_l = \sigma(b + W_l \cdot l) \tag{20}$$

$$h^\tau = h_l \odot h \tag{21}$$

(F) Output Fusion. This approach builds upon all the previously referenced papers and the proposed decoder combination described in 3.4. Here, a contextual location representation is learned using the same method described in Equation 17, but then it is concatenated to the output of the recurrent network, which is finally used as an input to the linear layer that produces the final ranking.

3.6 Implementation and Training

The proposed models are implemented in Pytorch [27] and are trained using the session-parallel mini-batch strategy introduced in [16]. The process starts with an ordering of the sessions and then a mini-batch is constructed by treating the first action of each session (x_1) as the input and the next one (x_2) as the target. In the next step the later becomes the input and x_3 becomes the target. If a session ends, the next session is put in its place and the respective hidden state is reset. Figure 4 shows an overview of how this process works.

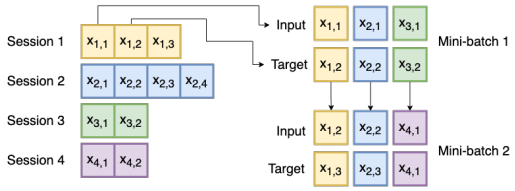

Figure 4: Session-parallel mini-batch training. Adapted from [16]

3.6.1 Loss Function. Learning can be done using several learning to rank loss functions, the most popular of which are the pair-wise losses Bayesian Personalized Ranking (BPR) [31] and TOP-1 [16]. In this work we use TOP-1 as it outperforms BPR in all of the cases. TOP-1 is defined as follows:

$$L_{TOP1} = \frac{1}{N_S} \cdot \sum_{j=1}^{N_S} \sigma(\hat{r}_{s,j} - \hat{r}_{s,i}) + \sigma(\hat{r}_{s,j}^2) \tag{22}$$

Where $\hat{r}_{s,i}$ is the ranking score of the relevant item (target), $\hat{r}_{s,j}$ the score of the negative sample and σ the sigmoid function. The loss is a regularized approximation of the relative rank of the relevant item. It aims to push the scores of the negative samples towards zero using the regularized term $\sigma(\hat{r}_{s,j}^2)$.

3.6.2 Negative Sampling. The proposed architectures output a ranking over all the items in the dataset. Calculating the loss function for all of them would make it non-scalable and as a result non-practical. To combat this, we treat the items of the other sessions in each mini-batch as the negative samples and only calculate the loss using those. In practice, this is essentially a popularity based sampling approach, since the more popular an item is the more likely it is to be present in a mini-batch. Moreover, we experimented with appending additional samples from the whole dataset, but this lead to the model overfitting on the training set.

4 EXPERIMENTS

In this section we go through the experimental setup. First, we define the research questions (4.1) and describe the datasets used to answer them (4.2). We then describe the evaluation metrics (4.3) and the baselines used for comparison (4.4). Finally, we go through the configuration details of the compared models (4.5).

4.1 Research Questions

We evaluate the proposed architectures by carrying out experiments on a dataset gathered from OLX[1], a leading peer-to-peer e-commerce website. With those experiments our purpose is to answer the following research questions:

(RQ1) Do the proposed parallel recurrent architectures outperform the current, state of the art methods in terms of Recall and MRR?

(RQ2) Does modeling location with a latent representation perform better than using parallel networks?

(RQ3) What is the most effective way of representing location? Is the proposed encoding useful, or a distance based approach suffices?

(RQ4) What is the most effective way of combining the independent parts of the parallel networks? Are multiplicative relationships effective, or an average approach works best?

4.2 Data

The data used to conduct experiments and evaluate the models was extracted from OLX impression logs, which contain detailed information over multiple user interactions on the website. Specifically, the system detects and logs a session as a sequence of actions happening in a single visit to the website. A session can be initiated via two funnels: either by typing a query to the search bar, or by browsing categories/items directly from the homepage. Three different types of interactions are collected: (a) impression: the item was shown to the user, (b) click: the user clicked on the item, (c) contact: the user contacted the seller. Multiple context attributes are collected for each event; we keep only those that are important for our task. Specifically, we extract the session and item identifiers, the timestamp of each action, the funnel and interaction type and the geolocation context (coordinates, city, district) of both buyer and seller.

For these experiments, impressions on listings were gathered from OLX during a seven day period. The dataset contains in total 591,058 sessions of ~40 million interactions on 438,075 unique items. We extract only the click interactions, filter out very short or very long sessions and remove geolocation based outliers. Finally, we exclude items which are clicked less than 10 times in the given period. After the filtering we are left with 55,128 sessions of 245,724 events on 12,973 unique items. Each session has an average length of ~4.5 events and the distance between buyer and seller is on average 261 kilometers. All the clicks for the first six days were used for training/validation and the ones on the seventh for testing (~20% of the data). Sessions and in-session clicks are all sorted by time before the training process begins.

[1] https://www.olx.com/

4.3 Evaluation Metrics

To evaluate the proposed models and compare them to the baselines, we use the common ranking evaluation metrics recall@k and MRR@k. The idea behind these choices is that the items most likely to be clicked by a user should appear on the top of the recommendation list [16, 29].

- **Recall@k**: defined as the percentage of cases in which the next clicked item appears in the top-k items of the recommendation list. Chosen because it is well correlated with click-through rate (CTR) [16, 23].

$$Recall@k = \frac{\#\ \text{rank}(x_\tau) \leq k}{\#\ \text{all cases}} \tag{23}$$

  where $rank(x_\tau)$ is the rank position in the recommendation list of the actual next clicked item in each session.
- **MRR@k**: Mean Reciprocal Rank, the average of the reciprocal ranks of the next-clicked items across all sessions. Important for assessing the quality of the order of the list.

$$MRR = \frac{1}{S}\sum_{i=1}^{|S|}\frac{1}{rank(i, x_\tau)} \tag{24}$$

  If $rank(i, x_\tau) > k$ the value of MRR is set to zero. We measure both Recall and MRR at the 10 and 20 first results in the recommendation list.

4.4 Baselines

We compare the models against (a) commonly used traditional baselines, (b) the basic RNN model and (c) devised location-based baselines.

- **POP**: Commonly used baseline for recommender systems. Always recommends the most popular item in the whole dataset [14].
- **S-POP**: Inspired by POP. Recommends the most popular item in the current session, while using the global popularity as a tie-breaker [16].
- **Item-KNN**: Common item-to-item recommendation baseline used to evaluate sequential models. It defines the similarity between two items as the number of inter-session co-occurences divided by the square root of the product of the number of sessions each item occurs. It is being regularized to avoid similarities by chance [9, 22].
- **LOC-POP**: Devised by us for this task. Gets the items most nearby the user and then recommends the most popular of those.
- **BaseGRU**: The model introduced by [16] (GRU4Rec), with the best configurations for the OLX dataset.
- **LocGRU**: Devised by us for this task. A model similar to BaseGRU, but takes as input only location information.

4.5 Implementation Details

The models were evaluated on their best configuration for the OLX dataset, using a combination of random search and manual adjustments on a validation set. The best models are then trained on both these datasets and evaluated on the test set. The hidden size of the recurrent unit was set to 500 and the batch size to 32 or 64. Adam was used to train the base and the parallel networks, while Adagrad performed best for the latent networks. The learning rates were set to 0.001 and 0.01 respectively. Finally, for the latent architecture the context size was set to 32 and the embedding size to 512. The best location representation is analysed in detail in Section 6.1.

5 RESULTS

The results of the baseline and the proposed architectures are presented in Table 1. As the table shows, the hybrid models outperform all non-sequential recommendation baselines by a large margin in terms of Recall and MRR on different ranks. They also overcome the RNN baseline by a smaller margin. Below we discuss the results of each different architecture.

5.1 Parallel Recurrent Results (RQ1)

Similarly to [16], the RNN approaches significantly outperform Item-KNN and all other commonly-used baselines. The recurrent units are capable of learning a comprehensive dense representation of the previous history of user interactions, which traditional approaches ignore. The results for the naive network that only uses location to predict the next item (LocGRU) show that there is value in modeling this location information, as it outperforms most non-sequential baselines. However, it also is far behind when compared to the baseline recurrent network, proving that modeling this information alone does not suffice for a reliable predictor and hybrid approaches should be used instead.

Modelling location in parallel networks seems to generally improve the basic sequential recommender, by enabling the network to learn representations of different attributes of a session simultaneously, but with the same end goal. In general, all proposed architectures outperform the baselines. Similar to the results in [17], we show that considering additional contextual signals does provide extra capacity to the sequential recommender and therefore improves its recommendations.

Out of the individual parallel architectures, the worst performing one is that which combines the two networks in the hidden state level (A). This indicates that, despite the hidden states holding information from the same previous sessions, the fact that they model it using different kinds of data makes them work independently, therefore they are not ideal candidates to be combined to produce a single output. The hidden model only increases the Recall by 2.2% and the MRR by 2% on average compared to BaseGRU. Similar behavior is observed in the encoder model (B), which contributes to identical increase in the Recall, while it does a better job in ranking relevant items higher, recording an improvement of 3.2% in MRR. On a similar notion, the outputs combined by this network are results of recurrent layers that model different parts of the session, therefore not ideal to give a joint ranking. The best performing parallel architecture is the one that does not fuse parts of the representation layers, but instead combines the next item rankings produced by the two networks independently (C). This is something to be expected, since we previously showed that there is some predictive capability in location information, even when modelled on its own (see LocGRU). Specifically, the decoder combination network improves the Recall by 3.3% and the MRR by 4.1% on average.

Table 1: Experimental Results on the OLX Dataset. Values are in percentages.

	OLX Dataset					
	Recall			MRR		
Models	@5	@10	@20	@5	@10	@20
POP	0.26	0.49	1.03	0.17	0.20	0.24
S-POP	0.14	0.35	0.83	0.05	0.08	0.11
LOC-POP	0.53	1.02	2.03	0.29	0.36	0.44
Item-KNN	6.04	10.20	15.15	3.56	4.20	4.56
LocGRU	5.20	7.89	11.46	2.89	3.24	3.48
BaseGRU	12.90	18.25	24.61	7.45	8.10	8.60
(A) Parallel Hidden	13.02	18.58	25.11	7.51	8.31	8.84
(B) Parallel Encoder	13.12	18.73	25.15	7.66	8.41	8.85
(C) Parallel Decoder	13.30	**18.92**	**25.62**	**7.73**	**8.48**	**8.94**
(D) Embedding Fusion	10.83	14.70	18.44	6.35	6.87	7.13
(E) Hidden Fusion	12.76	18.26	24.52	7.33	8.06	8.49
(F) Output Fusion	**13.32**	18.80	25.20	7.70	8.42	8.86

Bold face indicates best result in the corresponding metric.

5.2 Latent Embedding Results (RQ2)

Similar to the results in 5.1, the latent architectures significantly outperform the non-RNN baselines and in some cases the BaseGRU baseline. This indicates that there is value in learning by using location information to enhance the base network, rather than modelling it independently.

The fusion in the event embedding stage right before the recurrent unit (D) is the weakest of the proposed architectures, as it falls behind even when compared to BaseGRU. In terms of Recall, it is on average 20% worse than the baseline, while it falls short by 16% in terms of MRR. This indicates that, contrary to [21], contextualizing an event using location in the input level is not sufficient to create a comprehensive representation that encompasses both attributes at the same time. Using location to provide context to the hidden state instead yields better results, but still fails to outperform the RNN-baseline, recording almost similar results both in Recall and MRR. While it appears more useful to contextualize the hidden state instead of the input, the network still fails to explicitly model the context better than the simple behavioural pattern captured by the recurrent baseline. Similarly to the parallel architectures, the best way to inject the location context is after the recurrent unit, in this case on the output (or encoder) level (F). This method outperforms the baseline by 2.9% and 3.5% in Recall and MRR respectively. This further confirms that the base recurrent unit does a great job in capturing the sequential dependency of the user interactions, and further context becomes useful only when injected after the recurrent phase and before generating the final ranking.

6 ANALYSIS

In this section we analyse how the models behave with respect to different location representations (6.1) and different combination techniques (6.2).

Table 2: Effect of different Location Representations in the Parallel Decoder architecture.

	Parallel Decoder Model					
	Recall			MRR		
Method	@5	@10	@20	@5	@10	@20
City Encoding	13.15	18.80	25.50	7.61	8.35	8.81
District Encoding	13.13	18.81	25.47	7.62	8.37	8.82
Distance	13.26	18.67	25.55	7.54	8.26	8.73
Full Concatenation	**13.30**	**18.92**	**25.62**	**7.73**	**8.48**	**8.94**

6.1 Analysis of Location Representations (RQ3)

We design experiments to evaluate the effect of the proposed location representations in the performance of the parallel and the latent hybrid models . The results are presented in Tables 2 and 3. Specifically, we compare the full concatenated representation proposed in Section 3.3 with each individual part when used as its own predictor.

As the results in Table 2 show, in parallel architectures, distance alone is not enough to create a model that effectively captures the location dependency in user interactions. While its scores indicate that it is a strong individual predictor, appending additional details regarding the city and the neighborhood provide a representation that is more useful for recommendations. This is in part due to the fact that recurrent neural networks are ideal in modeling sequences of discrete representations [5]; in this case the one-hot encodings of the city and the neighborhood.

The picture is a bit different when location is used to provide context to the recurrent unit, as the results in Table 3 suggest. Modelling distance alone manages to outperform the other representations both in terms of Recall and MRR. These results are aligned with those reported by [5, 21], for time, where time-deltas provided enough additional context to improve the recommender. As a consequence, we expect the city and district representations to not be valuable when modelled on their own, which is confirmed by

Table 3: Effect of different Location Representations in the Latent Output Fusion architecture.

	Latent Output Fusion Model					
	Recall			MRR		
Method	@5	@10	@20	@5	@10	@20
City Encoding	13.19	18.59	25.11	7.67	8.38	8.84
District Encoding	13.28	18.66	25.11	7.63	8.34	8.78
Distance	**13.32**	**18.80**	**25.20**	**7.70**	**8.42**	**8.86**
Full Concatenation	**13.32**	18.65	25.18	**7.70**	8.40	8.85

Table 4: Effect of different output combination strategies in the Parallel Recurrent architecture.

	Recall			MRR		
Method	@5	@10	@20	@5	@10	@20
Addition	11.68	15.93	20.78	6.82	7.38	7.72
Multiplication	11.53	16.11	20.79	6.57	7.18	7.50
Weighted Average	**13.22**	**18.92**	**25.62**	**7.73**	**8.48**	**8.94**

the results. Interestingly, we observe that at the first five ranks the concatenated representations performs just as good as the distance scalar, indicating that providing the complete information yields good results on the top part of the recommendation list.

6.2 Analysis of Combination Techniques (RQ4)

We conduct experiments to evaluate the effect of different output combination techniques in the performance of the parallel recurrent architecture. The results are presented in Table 4. We evaluate weight addition, multiplication and weighted average.

As the results indicate, the most effective technique is a weighted average of the parameters of the parallel models before the final activation, outperforming the others by a big margin. This is also in alignment with the results reported by [17] for text and image parallel networks. The simple weight addition technique used in [5, 30] as a less complex approach in capturing 2-way relationships underperforms in this recommendation task. This is something we expect after looking at the overall results in Table 1, where it is clearly indicated the predictive power of the clicks is bigger than that of the additional context. The location context specifically, does indeed provide extra useful information to the model, but is not of the same importance as the user interactions. Multiplying the weights yields similar results to adding them. Interestingly though, this method leads to worse performance even when compared to the respective multiplication performed in the latent context model (F). This indicates that, despite its prediction strength, the recurrent location unit is not ideal to be used to directly apply context to the basic network.

Overall, in terms of Recall, the weighted average approach performs on average 3.1% and 3.0% better over addition and multiplication respectively, while it also beats them in terms of MRR by an average of 1.1% and 1.3% respectively.

7 CONCLUSION AND FUTURE WORK

In this study we proposed two families of hybrid recurrent neural networks to provide location aware next-item recommendation, based on previous interactions and additional location information. We introduced different ways of representing and modeling location, namely distance, city and neighborhood based. We evaluated the proposed models by conducting extensive experiments on the interaction logs of a big peer-to-peer e-commerce platform. Experimental results show that the architectures outperform popular state of the art models in terms of Recall and MRR.

We furthermore explored how different location representations impact the quality of recommendations in both of the hybrid families. We concluded that the parallel networks' strength in modeling discrete relationships leads to better recommendations, the more additional context we provide them. At the same time, the latent context networks work best with continuous data provided in terms of distance and do not benefit from a full representation of the location context. In analysing the performance of different combination techniques, we found that, contrary to what literature suggests, multiplicative relationships do not always lead to better representations and a weighted average approach can work better instead.

One of the limitations of the parallel architectures is that they add additional complexity to the models, since they need to learn twice as many parameters as a single recurrent network. Additionally, for these experiments both models were trained simultaneously, without leveraging the more advanced training techniques, like residual and interleaving training. Improvements in both the performance and the training efficiency are to be expected by employing these techniques. Moreover, it should be mentioned that the evaluation of the aforementioned models was done in a relatively small dataset compared to the publicly available recommender systems benchmarks. Finally, readers should take into consideration the fact that, similarly to all datasets generated from user interactions, the logged clicks used in this study were influenced by an already present recommender system.

In the future, the architectures can further be evaluated by conducting experiments in multiple datasets and different domains. Using a bigger dataset can provide additional benefit to the latent architecture, by yielding embeddings of better quality. Furthermore, focus should be directed to implementing alternative training techniques, rather than training both networks at the same time. Finally, additional contributions can be proposed to the loss function or the sampling technique, by punishing items that are further away or less accessible to the user.

ACKNOWLEDGMENTS

I would like to express my gratitude to Pengjie Ren, for his valuable support and for helping me with new ideas and inspiration whenever I struggled and to Dmitri Jarnikov for entrusting me with this project, and for giving me the freedom to set my own direction. I would also like to thank the entire Naspers AI team for creating a welcoming and friendly working environment. A special thanks to the *Hardly Working* group, for all the memories we've built together this year and to my family for the continuous support in all my ventures.

REFERENCES

[1] Michael Auli, Michel Galley, Chris Quirk, and Geoffrey Zweig. 2013. Joint language and translation modeling with recurrent neural networks. (2013).

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[3] Trapit Bansal, David Belanger, and Andrew McCallum. 2016. Ask the gru: Multi-task learning for deep text recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 107–114.

[4] Jie Bao, Yu Zheng, and Mohamed F Mokbel. 2012. Location-based and preference-aware recommendation using sparse geo-social networking data. In *Proceedings of the 20th international conference on advances in geographic information systems*. ACM, 199–208.

[5] Alex Beutel, Paul Covington, Sagar Jain, Can Xu, Jia Li, Vince Gatto, and Ed H. Chi. 2018. Latent Cross: Making Use of Context in Recurrent Recommender Systems. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining (WSDM '18)*. ACM, New York, NY, USA, 46–54. https://doi.org/10.1145/3159652.3159727

[6] Chen Cheng, Haiqin Yang, Michael R Lyu, and Irwin King. 2013. Where you like to go next: Successive point-of-interest recommendation. In *Twenty-Third International Joint Conference on Artificial Intelligence*.

[7] Heeyoul Choi, Kyunghyun Cho, and Yoshua Bengio. 2017. Context-dependent word representation for neural machine translation. *Computer Speech & Language* 45 (2017), 149–160.

[8] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

[9] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, et al. 2010. The YouTube video recommendation system. In *Proceedings of the fourth ACM conference on Recommender systems*. ACM, 293–296.

[10] Mukund Deshpande and George Karypis. 2004. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)* 22, 1 (2004), 143–177.

[11] Hancheng Ge, James Caverlee, and Haokai Lu. 2016. Taper: A contextual tensor-based approach for personalized expert recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 261–268.

[12] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press.

[13] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 6645–6649.

[14] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 173–182.

[15] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 843–852.

[16] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).

[17] Balázs Hidasi, Massimo Quadrana, Alexandros Karatzoglou, and Domonkos Tikk. 2016. Parallel Recurrent Neural Network Architectures for Feature-rich Session-based Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16)*. ACM, New York, NY, USA, 241–248. https://doi.org/10.1145/2959100.2959167

[18] Sepp Hochreiter. 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6, 02 (1998), 107–116.

[19] Dietmar Jannach and Malte Ludewig. 2017. When recurrent neural networks meet the neighborhood for session-based recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*. ACM, 306–310.

[20] Stefan Kombrink, Tomáš Mikolov, Martin Karafiát, and Lukáš Burget. 2011. Recurrent neural network based language modeling in meeting recognition. In *Twelfth annual conference of the international speech communication association*.

[21] Yang Li, Nan Du, and Samy Bengio. 2017. Time-dependent representation for neural event sequence prediction. *arXiv preprint arXiv:1708.00065* (2017).

[22] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing* 1 (2003), 76–80.

[23] Qiwen Liu, Tianjian Chen, Jing Cai, and Dianhai Yu. 2012. Enlister: baidu's recommender system for the biggest chinese Q&A website. In *Proceedings of the sixth ACM conference on Recommender systems*. ACM, 285–288.

[24] Haokai Lu and James Caverlee. 2015. Exploiting geo-spatial preference for personalized expert recommendation. In *Proceedings of the 9th ACM Conference on Recommender Systems*. ACM, 67–74.

[25] Muyang Ma, Pengjie Ren, Yujie Lin, Zhumin Chen, Jun Ma, and Maarten de Rijke. 2019. π-Net: A Parallel Information-sharing Network for Shared-account Cross-domain Sequential Recommendations. (2019).

[26] Jarana Manotumruksa, Craig Macdonald, and Iadh Ounis. 2018. A Contextual Attention Recurrent Architecture for Context-Aware Venue Recommendation. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM, 555–564.

[27] 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).

[28] Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. 2017. Personalizing session-based recommendations with hierarchical recurrent neural networks. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*. ACM, 130–137.

[29] Pengjie Ren, Zhumin Chen, Jing Li, Zhaochun Ren, Jun Ma, and Maarten de Rijke. 2018. RepeatNet: A Repeat Aware Neural Recommendation Machine for Session-based Recommendation. *arXiv preprint arXiv:1812.02646* (2018).

[30] Steffen Rendle. 2012. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology (TIST)* 3, 3 (2012), 57.

[31] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*. AUAI Press, 452–461.

[32] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*. ACM, 811–820.

[33] Badrul Munir Sarwar, George Karypis, Joseph A Konstan, John Riedl, et al. 2001. Item-based collaborative filtering recommendation algorithms. *Www* 1 (2001), 285–295.

[34] Guy Shani, David Heckerman, and Ronen I Brafman. 2005. An MDP-based recommender system. *Journal of Machine Learning Research* 6, Sep (2005), 1265–1295.

[35] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved Recurrent Neural Networks for Session-based Recommendations. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (DLRS 2016)*. ACM, New York, NY, USA, 17–22. https://doi.org/10.1145/2988450.2988452

[36] Bartłomiej Twardowski. 2016. Modelling contextual information in session-aware recommender systems with neural networks. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 273–276.

[37] Shuai Zhang, Yi Tay, Lina Yao, and Aixin Sun. 2018. Next item recommendation with self-attention. *arXiv preprint arXiv:1808.06414* (2018).

[38] Yuyu Zhang, Hanjun Dai, Chang Xu, Jun Feng, Taifeng Wang, Jiang Bian, Bin Wang, and Tie-Yan Liu. 2014. Sequential click prediction for sponsored search with recurrent neural networks. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*.

[39] Andrew Zimdars, David Maxwell Chickering, and Christopher Meek. 2001. Using temporal data for making recommendations. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 580–588.