Time is of the Essence: a Joint Hierarchical RNN and Point Process Model for Time and Item Predictions[*]

Bjørnar Vassøy[†]
Department of Computer Science, Norwegian University of Science and Technology
bjorva@stud.ntnu.no

Massimiliano Ruocco[†‡]
Department of Computer Science, Norwegian University of Science and Technology
massimiliano.ruocco@ntnu.no

Eliezer de Souza da Silva[†]
Department of Computer Science, Norwegian University of Science and Technology
eliezer.souza.silva@ntnu.no

Erlend Aune[†]
Exabel AS
aune@exabel.com

ABSTRACT

In recent years session-based recommendation has emerged as an increasingly applicable type of recommendation. As sessions consist of sequences of events, this type of recommendation is a natural fit for Recurrent Neural Networks (RNNs). Several additions have been proposed for extending such models in order to handle specific problems or data. Two such extensions are 1.) modeling of inter-session relations for catching long term dependencies over user sessions, and 2.) modeling temporal aspects of user-item interactions. The former allows the session-based recommendation to utilize extended session history and inter-session information when providing new recommendations. The latter has been used to both provide state-of-the-art predictions for when the user will return to the service and also for improving recommendations. In this work we combine these two extensions in a joint model for the tasks of recommendation and return-time prediction. The model consists of a Hierarchical RNN for the inter-session and intra-session items recommendation extended with a Point Process model for the time-gaps between the sessions. The experimental results indicate that the proposed model improves recommendations significantly on two datasets over a strong baseline, while simultaneously improving return-time predictions over a baseline return-time prediction model.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; • **Computing methodologies** → *Neural networks.*

KEYWORDS

Recommender Systems; Point Processes; Recurrent Neural Network; Session-Based Recommendation

ACM Reference Format:
Bjørnar Vassøy, Massimiliano Ruocco, Eliezer de Souza da Silva, and Erlend Aune. 2019. Time is of the Essence: a Joint Hierarchical RNN and Point Process Model for Time and Item Predictions. In *The Twelfth ACM International Conference on Web Search and Data Mining (WSDM '19), February 11–15, 2019, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3289600.3290987

1 INTRODUCTION

Finding relevant information on the web is an increasingly challenging problem, as more and more information becomes available. Users trying to find a particular piece of content are likely to enter a state of *information overload*, where they are not able to interpret the available information efficiently. Today, recommender systems are applied ubiquitously to assist users in navigating large sets of web content (video, audio, books, products, etc). A guiding principle in those systems is to apply predictive modeling to infer users' preferences using their history of interactions with the system. In many cases, however, it is not possible to track a long history of interactions, instead, there is only information about recent *sessions* of interactions. Due to increased focus on data privacy over the last years, this situation is likely to become more prevalent. Session-based recommender systems address this issue by being capable of using information available in the sessions themselves. In particular, the predictive problem is cast as a *sequence prediction problem*, where a history with a sequence of actions and interactions taken by the users in a set of sessions are applied to predict the next k actions in a given session or a new session [5].

Return time prediction is another important challenge that has been explored using different strategies. The main challenge here is developing a time prediction model that uses previous sessions information to infer *when* the user is going to return or start a new session in a given application or service. This task is important for two reasons: 1.) Modeling user retention rate in web-services through their return time dynamics can offer deep insight into the service at hand, allowing the service decision-makers to better optimize their service to maximize time spent on the service. This is a valuable metric within web economy, which is fueled by

[*]This work was carried out at the Telenor-NTNU AI-Lab, hosted by the Department of Computer Science, Norwegian University of Science and Technology.
[†]All the authors contributed equally
[‡]This author is also affiliated with Telenor Research.

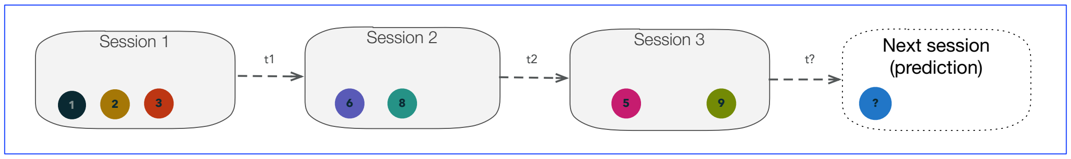


Figure 1: Representation of multiple sequential sessions with sequential item clicks and time between sessions

advertisement [10]. 2.) In the context of recommender systems, there is an interplay between temporal information and recommendations that can be exploited to harness predictive models and better understanding of users engagement with the recommender system [24].

In this article, we tackle both tasks jointly, based on the assumption that the interplay between users engagement with items and the temporal information is favorable in delivering good predictors for both tasks. We propose a new neural model capable of delivering next–item predictions using Recurrent Neural Networks (RNNs) and return–time predictions using Point Processes. For example, consider the context represented in Figure 1, where there are a sequence of sessions for a given user and within each session a series on user–item interactions (clicks, purchases, etc) with a series of items (represented by the colored circles). Here, there is also information about the time before the user returns to the service and start a new session. Given the motivation given above, it is valuable to build a model with capabilities for capturing inter- and intra sessions user–item dynamics and using the past user–item dynamics and the temporal information to predict when a new session would start, as well as deliver recommendations.

To summarize, the main contributions of this work are:

- Defining and introducing a Temporal Hierarchical Recurrent Neural Network (THRNN): a joint model for inter-session and intra-session recommendations and return-time prediction. It consists in three major components, a module for sequence of sessions of representations, a module for sequences of user-item interactions within each session, a module for the time interval between the sessions. The first two modules consists on a Hierarchical Recurrent Neural Network (HRNN) architecture and the third module is given by a Point Process which shares representations with the HRNN.
- A tuning mechanism in the training process that allows the time model to modulate the focus on short, medium or long time prediction. The mechanism consists of adding a control parameter in the loss function that allows us to control the importance of temporal information at training time.
- Experimental evaluation of the proposed model performance in both recommendation and return-time prediction tasks. The results show improvements in both tasks when compared to strong baseline models, and indicate that the joint modeling is particularly beneficial for the return–time prediction task. Finally for the return–time prediction task, we show that the control mechanism for short, medium or long time is effective.

1.1 Problem formulation

Given the set of user–item interactions; $S_{\text{train}} = \{(u, i, j, t_j) | u, i, j \in \mathbb{N}, t \in \mathbb{R}^+\}$, each tuple meaning that the user u interacted with the item i in a session j at time t_j. We aim to learn a function that estimates a score for new items within a session, a score for an initial recommendation in the next session, and the time gap prediction between the current and next session. More formally, given our training set S_{train}, and an index j for a position within the session, we want a function $f(S_{\text{train}}, j)$ that will output a tuple $(s_{j+1}, f_{j+1}, g_{t_{j+1}})$, with the intra-session recommendation score s_{j+1}, the next session initial item score f_{j+1} and the time-gap $g_{t_{j+1}}$.

Thus the objective is to build a model that can predict new items within a session and for the next session, as well as predict the time-gap for the next session (return–time prediction).

2 RELATED WORK

In recent years, numerous deep learning techniques have been successfully employed for recommendations. In particular the use of RNNs has been shown to be promising for session-based recommendation.

Session-based recommendation have been historically handled by item-to-item/ neighborhood-based methods [20] without considering the sequential nature of session data. A natural choice for modeling this sequential nature are RNN-based models such as those given in [5, 7, 16, 19, 22]. The GRU2REC architecture presented in [5] is widely credited to be the first to apply a RNN-based recommendation model and achieving state of the art results. The authors assume that the users with attached sessions are anonymous without any user-history, and assume that the sessions are independent of each other. Each session is modeled as a single layer of GRU units followed by a single feed-forward-layer. The GRU units, having recurrent connections, provide the state and captures the temporal dynamics of the sessions, while the feed-forward layer outputs the scores for each item. The model proposed was tested on two different datasets and achieved a 20-30% gain in the evaluated measures over [18], which is a nearest-neighbor-based model and was presented as the best performing baseline. A systematic comparison of multiple methods for session-based recommendation and the GRU2REC in [5] is presented in [14]. An extension of the GRU2REC model is presented in [19], with a second level of RNNs that tries to capture inter-session dynamics. The proposed model is a hierarchical RNN where one level considers inter-session dynamics and the other considers intra-session dynamics. As opposed to assuming fully anonymous users and totally independent sessions, this extension allows some simple and low-cost user history to be considered in order to provide better recommendations for live sessions. The user history is in the form of abstract representations of previous sessions. The inter-session RNN is fed a finite number of the most recent session-representations in chronological order, and the output is used as the initial hidden state of the intra-session RNN. The motivation behind this is to handle the cold-start problem. In contrast to [5], [19] achieved the best results when adding an embedding layer for the items. Additionally [19] did not use the interleaved session mini-batching scheme, instead opting for padded sessions. Nor did they sample the output or use any pairwise losses. The work in [16] proposes an architecture that is very similar to the model in [19]. They also propose an inter-session RNN layer in order to handle the cold-start problem when one has access to a limited user history in the form of previous sessions. The main difference between this hierarchical model and the one presented in [19] is that [16] only considered a session representation scheme based on the last hidden state of the intra-session RNN. Additionally, these session representations were created by passing the last hidden state to a final single layer feed-forward layer with a hyperbolic tangent activation function. They also experiment with propagating output of the inter-session network to all time-steps in the intra-session RNN, which complicates the model somewhat, but achieve slightly better results for one of the datasets evaluated. The hierarchical model proved to be the best performing overall by significant margin. In [22], the approach proposed in [5] is improved by proposing a data-augmentation pre-processing step for improving the robustness of the model as well as proposing to output an item embedding, instead of the individual scores for each item, to make the recommendations faster. The work in [7] presents an approach to adapt a nearest neighbor method for session-based recommendation task by finding sessions that are similar to the live session at the first step using an item cosine similarity measure. For predictions they use a weighted sum of nearest neighbour recommendations and RNN recommendations.

Other notable works on using RNNs have focused on how to best handle context- and feature rich input like [6, 13, 21]. In [13] a simple intra-session RNN model is extended by training and evaluate using different sets of weights based on the context of the input. This context can for instance be time and date of different granularities, location, or weather at the time of an event. In [6] the focus is to feed feature-rich input in the RNN input. In [21], the authors propose a new class of Contextual Recurrent Neural Networks for Recommendation (CRNNs) taking into account contextual information in two different ways: by combining the context embedding and input embedding, similar to [23] in one case, and by incorporating it in the model dynamics in the other case. Attention mechanisms have also been used within the RNN setting to provide recommendations [11]. The idea here is to use an attention mechanism over hidden states to provide a better representation from which to predict subsequent items.

Even though RNNs are inherently able to capture some temporal dependencies, this is largely based on the order of the events in a sequence. Aspects like time gap between events or sessions, the season, the year, the weekday, and the time of the day, are all temporal aspects that may influence the ideal recommendation in many domains, but which cannot easily be captured by the order of events alone. There have been attempts in both making time-aware recommendation by directly feeding such information in the RNN for the simple recommendation task [13], as well as modeling time for the task of predicting the return time of the next session [2, 8, 26].

The model proposed in [2] – based on a single layer RNN – attempts to predict the return-time of a user in addition to recommend the next item. The time prediction is modeled as a marked point process with intensity function conditioned on the history and the time to the next event. Similarly in combining recommendation and time-modeling, the model proposed in [8] use survival analysis to model the time instead of a marked point process. Another difference is that this model is used for next-basket recommendation task, and not just a single sequence of items. Furthermore, the time modeling is on the inter-basket/session time-gaps, and not the time between each selection. It means using a single level RNN for inter-session modeling directly, in contrast to [19] and [16], both implementing some form of inter-session modeling in one of the two levels of their hierarchical RNN models. The final model's recommendation capabilities was compared with two factorization based baselines and one based on neural networks. It was shown to outperform all of these on two different datasets. The time prediction was compared with expressive point processes, one of which was a Hawkes point process, and achieved smaller time errors than all of these. Finally, in [26], starting from the observation RNN units are good at modeling orders of entities, but does not offer any inherent support of time intervals between the entities, they proposed to incorporate the temporal aspect in the LSTM through explicit time gate.

Also, there has been efforts in using Point Process intensity modeling as an inspiration for hybrid Recurrent Neural Networks capable of time modeling. The approach in Du et al. [3] consist in creating a new type of recurrent neural unit with two outputs, one for the time model and another one for the general prediction model. The main contribution here is the proposal of a modeling approach for the intensity function using a combination of the hidden layers for long time dependency and recent time prediction, and combination the usual prediction loss with the negative log-likelihood of the Point process based on the intensity function. This results in a model capable of predicting markers and time of the markers, with an intensity function for the time model that is learned via a neural architecture. In Xiao et al. [25] a similar approach is taking, modeling the intensity function of a Point process via neural architecture, however in this case two RNN's are employed: one for time-series data, capturing background intensity rate, and another one for event data, capturing long range relationship between events. Finally, Mei and Eisner [15] is proposing a continuous time LSTM using Hawkes process as a starting point and making changes in the internal dynamics of the LSTM to achieve that goal. The resulting model is a LSTM with similar properties as the Hawkes process.

3 THE JOINT MODEL

We propose a system that aims to predict the return time for the next session and to recommend the next item. It is based on a hierarchical RNN (HRNN) model enhanced with a time model part. The inputs of the HRNN are the representations of previous sessions together with contextual information, followed by the item representations in the session. Figure 2 shows an overview of the system.

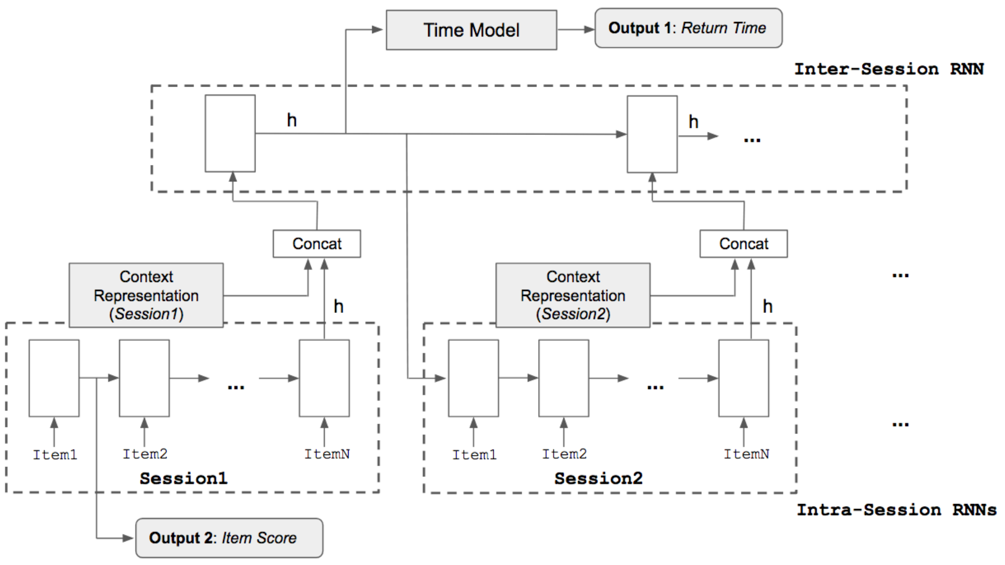


Figure 2: The intra-session RNN

The HRNN is inspired by [19], and consists of an *inter-session RNN* and an *intra-session RNN*. GRU units are used in both the intra-session and the inter-session RNNs. GRU was chosen due to its ability to remedy the vanishing gradient problem and since it was found to work better than LSTM for this problem and model structure. Like in [19], the inter-session RNN is fed with a fixed number of preceding session-representations. Moreover, in the proposed model, relevant session-related contextual information is concatenated to this input.

The final hidden state of the inter-session RNN is propagated to the intra-session RNN mainly representing the inter-session information, as well as employed for the return-time prediction task. The intra-session RNN uses the last hidden state of the inter-session RNN as initial state, and item representations in input. For each item embedding in the input, the corresponding output of the RNN is passed to a linear layer which outputs scores for each target items. The recommendation is given by selecting the items with highest scores.

3.1 Context Representation

Additional session-related contextual information, is concatenated with the last hidden state of the intra-session RNN, representing the input of the inter-session RNN. We consider three types of embeddings used in the main setup of the model: 1) Item embeddings, 2) inter-session gap-time embeddings and 3) user embeddings. The purpose of such embeddings is to learn finer dynamics and representations of the embedded entities. For instance, if two different artist often are listened to by users with similar tastes, an embedding layer would most likely learn artist representations that in some sense are similar to each other. By using RNNs, such representations can learn temporal dynamics in addition to a more general "similarity measure". A simplified example of such representational knowledge could be: "Artist A is almost always listened to before Artist B". Embeddings can also learn dissimilarity and non-linearity. For instance when thinking about gap times between sessions, there might be a higher correlation between gap sizes of 24 and 48 hours (periodic daily behavior) than between 12 and 24 hours.

3.1.1 *Item embeddings.*

. These embeddings represent each unique item in the dataset. The learned embeddings are directly handled by the intra-session RNN and consequently only trained by the resulting loss of this part of the model. This means that the item embeddings will affect earlier parts of the network, but then as input, not computational graphs. Hence, their gradient are unable to flow back to the embedding layer, and cannot be used to train the embeddings further. Due to the large number of items present in the considered datasets, the dimensionality of this embedding is the largest by far.

3.1.2 *Inter-session gap-time embedding.*

These embeddings represent the time gaps between sessions. The time-gap is first normalized and then divided into discrete buckets. The resulting bucket IDs can then be used to index embedding tables/layers to propagate the corresponding embedding. Two different normalization schemes were examined. Both are first given an upper bound, which sets a threshold of the gap time after which we don't consider the user active enough to be provided accurate time predictions. The gap-times that are greater than this bound is set to the upper-bound. The first normalization scheme, divides the gap-time range in uniformly large buckets. The benefit of this is that all the values in the gap-time range will belong to a bucket of equal size, causing no gap-time to be in the same bucket as a much higher or lower gap-time. A disadvantage of using this is that the earlier "popular" buckets can be overcrowded and the later ones can end up being almost empty, which can make such embeddings hard to train. One also needs a high resolution to cover the finer differences in the smaller time-gap ranges, which further increases the problem of sparse buckets. In the second normalization scheme, the gap times are first transformed with a log function, before the transformed range is divided uniformly into buckets. This results in a more evenly distributed number of gap times in the different buckets, but at the cost of cruder resolution for larger gap-times where the corresponding buckets cover much more time than the earlier ones. We observed that the second scheme is performing better with small resolutions, but was overall out-performed by the uniform scheme with higher resolution. Since having high resolution is a non-issue, both with regards to model performance and run-times, the uniform was deemed the better option.

3.1.3 User embedding. These embeddings are mainly inspired by [9], and learning the user behaviour beyond the history of session representations is especially useful with long user histories. For instance, if a user behaves a bit unusual and sporadic in the last few sessions, a model with user embeddings can have information about long term user behavior, which can help making sense of/override the recent noisy behavior.

3.2 Time Model

The main goal of this model is to predict the time until the next session start, given a fixed length history in the form of abstract session representations and corresponding contextual temporal information. The time modeling is heavily inspired by the work in [3], where time modeling is used to both predict the the time of the next item recommendation given a single sequence of previous items, as well as to improve the recommendation. In their model, the gap-times between selected items are considered to be drawn from a marked point process. The parameterization of the marked point process is defined by the authors and is dependent on the previous selection history modeled as a RNN, with corresponding inter-selection gap-times, and on the time of the last selection. In our system the history is based on session representations and not individual items. Consequently, the corresponding time-gaps are the times between the session representations in the history. This inter-session modeling, as well as the concatenated embeddings, is more similar to what was done in [9]. In the point process, we define the intensity function as:

$$\lambda^*(t) = \exp(v^{t\top} \cdot h_j + w^t \cdot g_j + b^t) \tag{1}$$

where h_j is the j-th hidden state, $g_j = t - t_j$ - the time since the last session (in which t_j is the last timestamp in the last session and t is the time variable), v is a vector of weights with the same dimensionality as h_j, w^t is a single weight and b^t is a bias term. $v^{t\top} \cdot h_j$ comprises the historical influence on the intensity function, while $w^t \cdot g_j$ is the current influence. The full conditional density function is then defined as follows:

$$f^*(t) = \lambda^*(t)\exp\left(-\int_{t_j}^{t} \lambda^*(\tau)d\tau\right) \tag{2}$$

finally, where the intensity Equation 1 has been substituted into the conditional density function Equation 2, the full expression of the marked point process is:

$$f^*(t) = \exp\left\{v^{t\top} \cdot h_j + w^t \cdot g_j + b^t + \frac{1}{w^t}\exp(v^{t\top} \cdot h_j + b^t) - \frac{1}{w^t}\exp(v^{t\top} \cdot h_j + w^t \cdot g_j + b^t)\right\} \tag{3}$$

The expected return time $\hat{t}_{j+1}$ is computed as the expected value in form of weighted area over the probability distribution as follows:

$$\hat{t}_{j+1} = \int_0^\infty t \cdot f^*(t)dt \tag{4}$$

The integration of the density distribution of the point process (Equation 3) does not have an analytic solution. Thus, prediction has to be approximated by numerical integration. To handle the infinite upper integration bound, a simple upper cut-off time is defined. While this is another approximation, the approximation becomes negligible when setting the cut-off time a bit higher than the vast majority of gap-times found in the data.

3.3 Loss

Creating a loss function is a matter of combining a recommendation loss and a return time loss. The latter is simply the log-likelihood of the time-gap point process distribution. On top of that, in order to control and tune the *importance* of the short, medium and long term data points in the training process, we changed the loss by adding the exponent $\alpha \in (0, 1)$ to the time-step variable g_j. The addition of the exponent on the time-step variable comes from looking at the gradient of the negative log-likelihood in relation to w^t: $-\frac{\partial \log f^*(t)}{\partial w^t} = -g_j + \frac{c_1}{(w^t)^2}\exp(g_j w^t)(g_j w^t - 1) + c_2$ (c_1 and c_2 are fixed terms not depending directly on the time-interval g_j or the weight w^t). The gradient has two basic components related to the time-gap information, a linear component coming from the current session and a exponential component that is related to the past interactions in the Point process model (see discussion of Equation 1). This led to the conclusion that a simple mechanism to modulate the emphasis between shorter time prediction or longer time prediction is to exponentiate the time-interval variable g_j, since this operation would affect the linear and exponential part differently. Another way to see the same intuition is to observe that the negative log-likelihood is approximately linear on small time-gaps and exponential on large time-gaps, meaning that exponentiating the time-gaps to a number closer to zero would make the shorter time-gaps dominate the loss more than the larger time-gaps. This addition proved to be valuable for tuning trade-offs between short-medium-long time predictions as we will see in Section 5. The final time loss is:

$$L_{time}(g_j, h_j, w) = -\Big(v^{t\top} \cdot h_j + w^t \cdot g_j^\alpha + b^t + \frac{1}{w^t}\exp(v^{t\top} \cdot h_j + b^t) - \frac{1}{w^t}\exp(v^{t\top} \cdot h_j + w^t \cdot g_j^\alpha + b^t)\Big) \tag{5}$$

The recommendation loss, $L_{rec}(s_{j+1}, i)$, is simply the softmax for item i given session representation at time $j + 1$, i.e. s_{j+1}. We combine these losses using a weighted mean:

$$L_{total} = \alpha L_{time}(g_j, h_j, w) + \beta L_{rec}(s_{j+1}, i) \quad (6)$$

Here, i is the target item of the intra-session recommendation. s_{j+1} is the score of the intra-session recommendations and g_j is the target time-gap.

4 EXPERIMENTAL SETTING

In order to ensure the reproducibility of the experiments, and for further implementation details, we make our code available on a github repository [1].

4.1 Datasets

The evaluation is performed on two different datasets as in [19]: the *LastFM* dataset [1] containing listening habits of users on the music website Last.fm and the *Reddit* dataset [2], on user activity on the social news aggregation and discussion website Reddit. Last.fm is a music website where users can keep track of the songs they listen to as well as sharing this with their peers. The data is in the form of tuples containing *user-id*, *artist*, *song* and *timestamp*, each representing a single listening event. Reddit is a popular forum/discussion website, where people can share and comment on different news, creations, pictures and other topics. Its structure is divided into different sub-forums, named subreddits, which define the topics/interests/allegiance of the posts to be posted there. This data is in the form of tuples containing a *user*, a *subreddit* and a *timestamp*, and each of these represents a single event where the user has commented on a post within the specific subforum at the given timestamp.

4.2 Data Preprocessing

The data were first preprocessed by removing noisy and irrelevant data and defining markers in the data following the steps in [19]. The Reddit dataset contains a log of user interaction on different subreddits (sub-forums), with timestamps. Here, an interaction is when a user adds a comment to a thread. Since the dataset does not split the events into sessions, we did this manually by specifying an inactivity time limit. Using the timestamps, we let consecutive actions that happened within the time limit belong to the same session. That is, for a specified time limit Δ_t, and a list of a user's interactions $\{a_{t_0}, a_{t_1}, \ldots, a_{t_n}\}$, ordered by their timestamps t_i, two consecutive interactions a_{t_i} and $a_{t_{i+1}}$ belong to the same session if $t_{i+1} \leq t_i + \Delta_t$. We set the time limit to 1 hour (3600 seconds) for the LastFM and 30 minutes (1800 seconds) for the Reddit dataset as in [19]. For both dataset we then removed all L consecutively repeating items, reducing them to only one occurrence. Following [19] we set the maximum length, L, of a session to $L = 20$, split sessions that had a length $l > L$ into two sessions and removed sequences of length more than $2L$. As [19], we also simplified the LastFM dataset by ignoring the specific song of each user interaction and only use the artists. When modeling the inter-session time-gaps, the sessions that are split because their length are not considered separate sessions, since this would introduce noise. We solved this problem by setting the last timestamp in the first half and the first timestamp in the second half, to be the start time of the full session, resulting in a gap-time equal to 0. The contribution will then be masked away from the time loss, making sure the model does not to train on these gap-times. Finally, the datasets were split into a training set and a test set on a per user basis. Each user's sessions were sorted by the timestamp of the earliest event in the session, and the earliest 80% of his sessions were placed in the training set, while the remaining 20% of the sequences, that contain the most recent sessions of each user, were allocated to the test set. Table 1 shows statistics for the two datasets after preprocessing (before splitting into training and test sets).

	Reddit	Last.fm
Number of users	18.271	977
Number of sessions	1.135.488	630.774
Sessions per user	62,1	645,6
Average session length	3,0	8,1
Number of items	27.452	94.284

Table 1: Statistics for the datasets after preprocessing

4.3 Baselines

In order to validate the performance of our model for both the tasks of return session time prediction and next item recommendation the THRNN model is compared to the following baselines, in addition to the intra-session Hierarchical RNN itself (HRNN). These baselines have been shown to be the strongest ones for such tasks as demonstrated in [16, 19] for the session based recommendation and [2] for time prediction.

GRU4REC. [3]: implementation based on the seminal work of Hidasi et al. [5] that uses session-parallel mini-batch training process and ranking-based loss functions for learning a model using solely the items sequence information within a session.

HRNN. The Hierarchical RNN (also Inter- and Intra-Session RNN), is a simpler version of the architecture used in this article. Namely, take the architecture presented in Figure 2, remove the time model and the user embeddings. The resulting model is the one given in [19]. The HRNN baseline parametrization is the same as in this article where it has been shown to otuperform the GRU4REC and other strongest baselines, such as Item K-NN [12] and BPR-MF [17]

Hawkes process. The Hawkes process is a self-exciting Point process with intensity given by

$$\lambda(t|\mathcal{H}_t) = \gamma_0 + \alpha \sum_{t_j \in \mathcal{H}_t} \gamma(t, t_j) \quad (7)$$

for some constants γ_0, α, a kernel γ and the history of events $\mathcal{H}_t$ [4]. γ_0 defines a baseline intensity, while γ defines the propensity with which the intensity is excited by events in the process itself. The kernel typically has the property that if a number of events happen with short time gaps, the intensity λ becomes larger than if the same

[1]https://github.com/BjornarVass/Recsys
[2]Subreddit interactions dataset: https://www.kaggle.com/colemaclean/subreddit-interactions
[3]https://github.com/hidasib/GRU4Rec

	Reddit	Last.fm
Item Embedding size	50	100
User Embedding Size	10	10
Time-Gap Embedding Size	5	5
Learning rate	0.001	0.001
Learning rate-time	0.0001	0.0001
Dropout rate	0	0.2
Max. recent session representations	15	15
Mini-batch size	100	100
Number of GRU layers, intra-session level	1	1
Number of GRU layers, inter-session level	1	1

Table 2: Best configurations for the RNN models.

number of events happen with larger gaps. The history of event $\mathcal{H}_t$ has the list of event times up to time t, $\{t_1, ..., t_n | \forall j \leq n : t_j < t\}$. For the baseline, we fit one Hawkes process for each user (using the last 15 sessions), and set the kernel to an exponential function. Return-time predictions are expected values of the Hawkes process primed by the time gaps of the 15 most recent sessions for each user. We also used a *Hawkes long-term* fitting procedure, for which the entire per-user training dataset is used for estimating the parameters in the Hawkes process.

4.4 Evaluation Metrics and Hyper-Parameters tuning

We used *Recall@k* and *MRR@k* with $k = 5, 10, 20$ to evaluate all models for the recommendation task and *Mean Absolute Error* (MAE), for the return-time prediction. In addition to the baselines already discussed, we also compared the THRNN with other models on the two presented datasets. We experimented with mini-batch sizes, embedding sizes, learning rate, dropout rate, using multiple GRU layers, and number of session representations to find the best configurations for each dataset. The best configurations we found are summarized in Table 2. *Learning-rate time* refers to the learning rate of w and v^t respectively, in the Equation 5. This value had to be reduced in order to stop this loss from diverging, mainly due to the exponential function as well as the scaling with the time-gap in the formula. We then find the best scaling factors for the loss function in Equation 6 to be $\alpha = 0.45$ and $\beta = 0.45$.

5 RESULTS AND DISCUSSION

5.1 Effect of parameter α

In order to show the effect of the parameter α in Equation 6, we evaluate the model by varying $\alpha \in [0.3, 0.5, 0.7, 0.9, 1.0]$ for the task of return time prediction. In Figure 3 we see performance, in term of MAE, of the proposed architecture for both LastFM and Reddit dataset. In both cases we observe that with α close to 1.0 the MAE decrease for longer time gap, showing an increase of accuracy in the time prediction for such gaps. In particular, for the LastFM dataset the performance of all the initializations seems to meet at 1.5 days time-gaps which is the middle point between the $[0.5 - 1.5]$ days interval and $[1.5 - 2.5]$ days interval. We can also observe that the plot looks more linear the smaller α is set. For example, for $\alpha = 0.3$ the plot is similar to the one produced by predicting an averaged time-gap for every single prediction. For the LastFM dataset, we see that by decreasing the value of α, the focus of the model is to have better performance on smaller time-gaps but unlike the Reddit dataset, on the long-term gap, the performance of the model tends to deteriorate faster. This is probably due to differences in distribution of the time gaps for the two datasets.

For both datasets, Figure 3 clearly illustrates the trade-off between error for the most frequent time-gaps and the effective prediction range of the model. Choosing α should be based on the service for which we are modeling return times. For some services, modeling shorter time gaps are more important that modeling longer ones.

5.2 Return time prediction: Long Term vs Short Term

In Figure 4, we illustrate the performance of the proposed model for return time predictions versus two baseline Hawkes models, with tuning parameter $\alpha \in [0.3, 0.5, 0.9]$ in Equation 5. For short time-gaps, the joint model of session embeddings and point process intensity gives better predictions than the baselines. The results are plotted alongside the number of observations for each time-gap, which gives insightful information about how the user–item interaction within a session is relevant for the model. We observe that the joint model outperforms the baseline consistently at time-gaps where there are more user–item observations. This is intuitive because of the shared learned parameters between the time model and the intersession model, in the sense that those parameters are not only fitting the time-gaps, but also the user–item interactions dynamics.

We see that tuning α allows us to control which time-gaps are most important to the model. Tuning for the most important time-gaps, however, comes at the cost of having worse return-time predictions for the time-gaps that are deemed less important.

5.3 Impact on actual recommendation

The experimental results for the recommendation task are summarized on Table 3 (for the LastFm dataset) and Table 4 (for the Reddit dataset). We observe an improvement in both metrics (Recall and MRR) at distinct levels (5, 10 and 20). This indicates that there is a win-win situation by incorporating intensity-based time modeling in the HRNN model, improving both recommendations and return time prediction when compared to respective baselines. However, it is worth noting that the difference between THRNN and HRNN is not just the time modeling, but also some added use of contexts in the THRNN model. It is also noticeable that the improvements of THRNN over HRNN, although significant, are less salient in absolute value than the improvements over GRU4REC. The simplest hypothesis is that the inter-session layer of HRNN is already capturing some of temporal dynamics between the sessions, for example it is possible that the latent representation are encoding time information correlated with the changes of user–item interactions from the end of one session to the beginning of the next session (for example if there are different profiles for distinct time-gaps of items that are typically accessed when finishing a session and items that typically are accessed in the beginning of the next session).

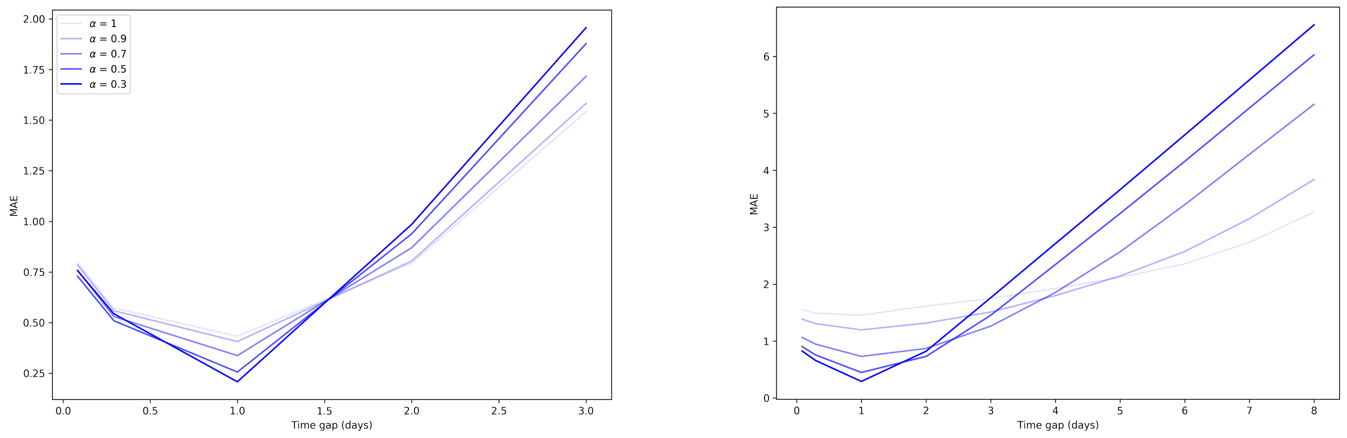
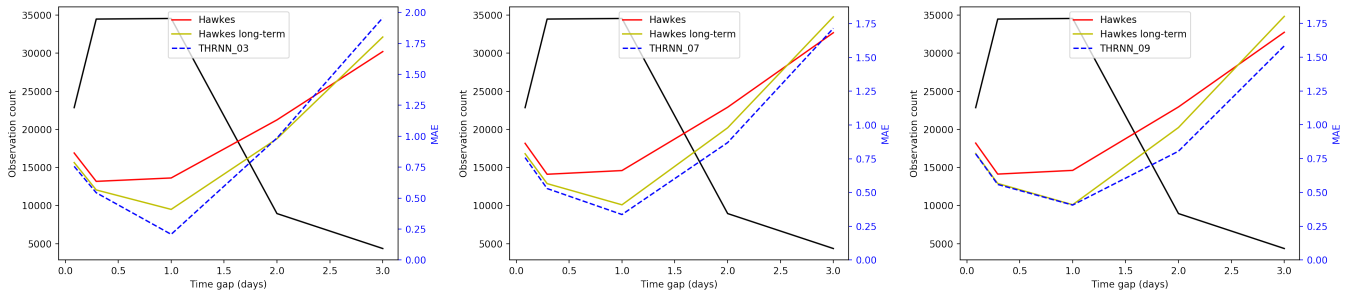
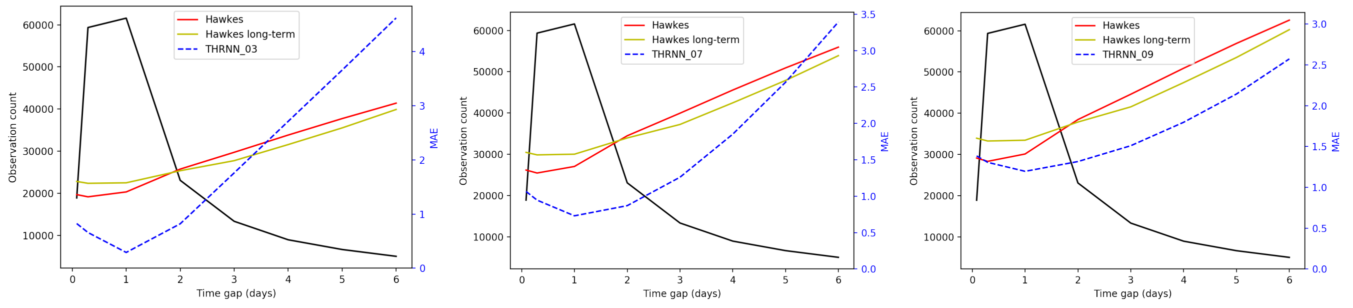


Figure 3: Time prediction MAE with different values of parameterα evaluated on LastFM dataset (left) and Reddit dataset (right). Runs with $\alpha = 0.1$ are not included since time-specific gradients diverged for this initialization



(a) LastFM dataset: α values of 0.3 (left), 0.5 (center), 0.9 (right)



(b) Reddit dataset: α values of 0.3 (left), 0.5 (center), 0.9 (right)

Figure 4: Time prediction MAE compared with Hawkes baselines with different values of parameter α.

6 CONCLUSION

In this article we introduced a new joint model capable of intra-session and inter-session recommendations and inter-session return time prediction. Our results indicates that jointly modeling of intra-session and inter-session user-item interactions, both for recommendations and time-predictions is favorable for the session-based items recommendations and return time prediction task. The joint model is competitive with the state-of the art in both the recommendation and time-prediction task. Particularly in the latter we report significant improvement when compared to state-of-the-art models for time prediction in online social networks (Hawkes point process models). Also, we introduced in the loss function a mechanism for controlling and trade-off short, medium and long time prediction accuracy and show in the results that it is effective. Future studies within this framework could, e.g., entail how we can fuse multiple losses, each focusing on different time-gaps, for a more holistic loss.

	R@5	R@10	R@20	MRR@5	MRR@10	MRR@20
GRU4REC	0.1349 ± 0.0004	0.184 ± 0.0002	0.2474 ± 0.0002	0.086 ± 0.0003	0.0925 ± 0.0002	0.0969 ± 0.0002
HRNN	0.1415 ± 0.0005 (+4.9%)	0.1993 ± 0.0007 (+8.3%)	0.2751 ± 0.0006 (+11.2%)	0.0876 ± 0.0004 (+1.8%)	0.0952 ± 0.0004 (+2.9%)	0.1004 ± 0.0004 (+3.7%)
THRNN	**0.1437 ± 0.0004** (+6.6%)	**0.2026 ± 0.0006** (+10.1%)	**0.2795 ± 0.0006** (+13.0%)	**0.0889 ± 0.0002** (+3.4%)	**0.0967 ± 0.0003** (+4.5%)	**0.102 ± 0.0003** (+5.3%)

Table 3: Table with the recall and MRR on the LastFM dataset.

	R@5	R@10	R@20	MRR@5	MRR@10	MRR@20
GRU4REC	0.3208 ± 0.0004	0.3959 ± 0.0005	0.475 ± 0.0003	0.2346 ± 0.0006	0.2445 ± 0.0006	0.25 ± 0.0006
HRNN	0.4432 ± 0.0013 (+38.1%)	0.5316 ± 0.0009 (+34.3%)	0.616 ± 0.0012 (+29.7%)	0.317 ± 0.0016 (+35.1%)	0.3288 ± 0.0016 (+34.5%)	0.3347 ± 0.0015 (+33.9%)
THRNN	**0.4468 ± 0.0013** (+39.3%)	**0.5366 ± 0.001** (+35.6%)	**0.6228 ± 0.0009** (+31.1%)	**0.3191 ± 0.0015** (+36.0%)	**0.3311 ± 0.0014** (+35.4%)	**0.3371 ± 0.0014** (+34.8%)

Table 4: Table with the recall and MRR on the Reddit dataset.

REFERENCES

[1] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. 2011. The Million Song Dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*.

[2] Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. 2016. Recurrent Marked Temporal Point Processes: Embedding Event History to Vector. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. ACM, New York, NY, USA, 1555–1564. https://doi.org/10.1145/2939672.2939875

[3] Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. 2016. Recurrent Marked Temporal Point Processes: Embedding Event History to Vector. In *KDD*. ACM, 1555–1564.

[4] Alan G Hawkes. 1971. Spectra of some self-exciting and mutually exciting point processes. *Biometrika* 58, 1 (1971), 83–90.

[5] 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

[6] 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. http://doi.acm.org/10.1145/2959100.2959167

[7] 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 (RecSys '17)*. ACM, New York, NY, USA, 306–310. https://doi.org/10.1145/3109859.3109872

[8] How Jing and Alexander J. Smola. 2017. Neural Survival Recommender. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining (WSDM '17)*. ACM, New York, NY, USA, 515–524. https://doi.org/10.1145/3018661.3018719

[9] How Jing and Alexander J. Smola. 2017. Neural Survival Recommender. In *WSDM*. ACM, 515–524.

[10] Komal Kapoor, Mingxuan Sun, Jaideep Srivastava, and Tao Ye. 2014. A hazard based approach to user return time prediction. In *KDD*. ACM, 1719–1728.

[11] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural Attentive Session-based Recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017*. 1419–1428.

[12] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon.Com Recommendations: Item-to-Item Collaborative Filtering. *IEEE Internet Computing* 7, 1 (Jan. 2003), 76–80.

[13] Qiang Liu, Shu Wu, Diyi Wang, Zhaokang Li, and Liang Wang. 2016. Context-aware Sequential Recommendation. *CoRR* abs/1609.05787 (2016). arXiv:1609.05787 http://arxiv.org/abs/1609.05787

[14] 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

[15] Hongyuan Mei and Jason Eisner. 2017. The Neural Hawkes Process: A Neurally Self-Modulating Multivariate Point Process. In *NIPS*. 6757–6767.

[16] 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 (RecSys '17)*. ACM, New York, NY, USA, 130–137.

[17] 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 (UAI '09)*. AUAI Press, Arlington, Virginia, United States, 452–461.

[18] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian Personalized Ranking from Implicit Feedback. *CoRR* abs/1205.2618 (2012). arXiv:1205.2618 http://arxiv.org/abs/1205.2618

[19] Massimiliano Ruocco, Ole Steinar Lillestøl Skrede, and Helge Langseth. 2017. Inter-Session Modeling for Session-Based Recommendation. In *Proceedings of the 2nd Workshop on Deep Learning for Recommender Systems, DLRS@RecSys 2017, Como, Italy, August 27, 2017*. 24–31.

[20] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based Collaborative Filtering Recommendation Algorithms. In *Proceedings of the 10th International Conference on World Wide Web (WWW '01)*. ACM, New York, NY, USA, 285–295.

[21] Elena Smirnova and Flavian Vasile. 2017. Contextual Sequence Modeling for Recommendation with Recurrent Neural Networks. In *Proceedings of the 2Nd Workshop on Deep Learning for Recommender Systems (DLRS 2017)*. ACM, New York, NY, USA, 2–9.

[22] 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

[23] Bartlomiej Twardowski. 2016. Modelling Contextual Information in Session-Aware Recommender Systems with Neural Networks. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16)*. ACM, New York, NY, USA, 273–276.

[24] Qingyun Wu, Hongning Wang, Liangjie Hong, and Yue Shi. 2017. Returning is Believing: Optimizing Long-term User Engagement in Recommender Systems. In *CIKM*. ACM, 1927–1936.

[25] Shuai Xiao, Junchi Yan, Xiaokang Yang, Hongyuan Zha, and Stephen M. Chu. 2017. Modeling the Intensity Function of Point Process Via Recurrent Neural Networks. In *AAAI*. AAAI Press, 1597–1603.

[26] 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 *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI'17)*. AAAI Press, 3602–3608. http://dl.acm.org/citation.cfm?id=3172077.3172393