JUMP: A Joint Predictor for User Click and Dwell Time

Tengfei Zhou[1], **Hui Qian**[1*], **Zebang Shen**[1], **Chao Zhang**[1], **Chengwei Wang**[1], **Shichen Liu**[2], **Wenwu Ou**[2]
College of Computer Science and Technology, Zhejiang University[1], Searching Group of Alibaba Inc.[2]
{zhoutengfei,qianhui,shenzebang,zczju, rr}@zju.edu.cn, shichen.lsc@alibaba-inc.com, santong.oww@taobao.com

Abstract

With the recent proliferation of recommendation system, there have been a lot of interests in session-based prediction methods, particularly those based on Recurrent Neural Network (RNN) and their variants. However, existing methods either ignore the dwell time prediction that plays an important role in measuring user's engagement on the content, or fail to process very short or noisy sessions. In this paper, we propose a joint predictor, JUMP, for both user click and dwell time in session-based settings. To map its input into a feature vector, JUMP adopts a novel three-layered RNN structure which includes a fast-slow layer for very short sessions and an attention layer for noisy sessions. Experiments demonstrate that JUMP outperforms state-of-the-art methods in both user click and dwell time prediction.

1 Introduction

session-based Prediction (SBP) is an emerging problem in recent recommendation system research. For services like e-commerce, media streaming, or classified site, previous user behavior sequence is usually available. In such circumstance, a session-based predictor is adopted to forecast user's next behavior, either the next *user click* or the *dwell time* (the amount of time that users spend on the clicked items).

SBP tasks can be solved by classical recommendation algorithms like item-to-item methods and matrix factorization based ones [Hidasi and Tikk, 2016; Linden *et al.*, 2003; Musto *et al.*, 2015]. Despite of their efficiency, such methods fail to exploit the session's sequential nature, which precludes their use in practical problems. Alternatively, many researchers resort to Markov Decision Processes (MDP) based technique [Shani *et al.*, 2005; Tavakol and Brefeld, 2014] for its capability in capturing the ordering information. A major issue for applying MDP is that the state space becomes unmanageable quickly when trying to include all possible sequences of potential user selection over all items.

Recently, deep learning methods have brought a tremendous potential to the area of recommender system. Of these methods, Recurrent Neural Networks (RNN), especially, Gated Recurrent Unit (GRU), have emerged as powerful methods for modeling sequential data in SBP [Hidasi *et al.*, 2015; Dallmann *et al.*, 2017; Hidasi and Karatzoglou, 2017; Zhang *et al.*, 2014; Wu *et al.*, 2016a]. Essentially, most of these methods can be formulated as learning to rank problems, which use ranking losses to achieve optimal choices.

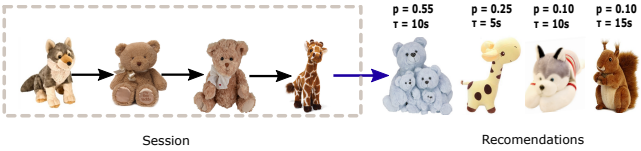

Figure 1: session-based Prediction (SBP) task. In SBP task, current sessions are used to perform recommendation.

Although remarkable progress has been made for SBP with RNN methods, current SBP techniques as a productivity lever are still challenging. (a) First, the prediction of dwell time is not taken into consideration. Existing studies have proved that dwell time is an important metric to measure user engagement on the content and should be used as a proxy to user satisfaction [Yi *et al.*, 2014; Yin *et al.*, 2013]. On the other hand, sophisticated neural networks as RNNs are prone to overfit due to the extreme sparsity of user clicks. It is illustrated that by coercing the RNN with an extra task, the risk of overfitting will plummet [Baxter, 1997; Ruder, 2017]. In SBP, the dwell time prediction is a natural choice. (b) Second, the input sequence may include many noisy or irrelevant parts, especially when the session is long. Most RNN based SBP solvers treat the information in the whole sequence equally, thus the noisy or irrelevant parts may grievously degrade the recommendation performance. (c) Third, existing RNN based SBP solvers are sub-optimal when the input sequence is too short. Many researchers observed that very short history interactions significantly deteriorate the GRU's performance [Quadrana *et al.*, 2017; Ruocco *et al.*, 2017]. Since short sessions comprise a significant portion of the practical situations, improving recommendation results for short sessions can be highly propitious.

To bridge these gaps, we propose a joint predictor, JUMP[1], for both user click and dwell time in the session-based set-

*Corresponding author

[1]JUMP is for Joint User-click and dwell tiMe Predictor.

tings. The main contributions are summarized as follows.

- Dwell time is modeled by *survival analysis*, which has sound underlying mathematical principles. Based on this, a joint predictor is designed for both user click and dwell time.
- A novel three-layered RNN structure is constructed to map the session input to feature vector. Instead of forcing it to encode all information into one fixed-length vector, we use an attention layer to allow the network to refer back to the input sequence, which improves the robustness against noise.
- We also use redesigned fast-slow cells to enhance the performance for very short sessions. In the meanwhile, Recurrent Highway sub-cells are embedded in the fast-slow cell to keep gradient flow stable.

In the experiments, we evaluate the performance of JUMP by recommendation tasks. Empirical results show that the proposed model has better performance in prediction of both user click and dwell time than the state-of-the-arts.

2 Relate Work

Recurrent Neuron Network The ordering nature of SBP makes it suitable to be formulated as a sequential learning problem. Highly successful in natural language modeling[Lin *et al.*, 2017; Sukhbaatar *et al.*, 2015], RNNs have become the most popular tool to approach such tasks. Traditional RNN architectures suffer from vanishing and exploding gradients [Bengio *et al.*, 1994], which renders the optimization difficult and prohibits RNNs from learning long dependency tasks. For a long time, LSTM has been the de facto way to address the gradient vanishing/exploding problem [Hochreiter and Schmidhuber, 1997]. Later GRU has been proposed as a simplification of LSTM using fewer gates, but with competitive performance [Chung *et al.*, 2014]. Recently, a new architecture named Recurrent Highway (RH) has been proposed [Zilly *et al.*, 2016]. Suarez suggests that RH has better gradient flow than both LSTM and GRU [Suarez, 2017]. Besides, the attention mechanism [Bahdanau *et al.*, 2014] has become an inseparable part of modern RNN structures, which significantly boosts the performance of RNNs, especially for long sequences. Additionally, a fast-slow RNN framework has shown that interconnecting different RNN cells and making them evolve at distinct speed are helpful to learn complex transition functions [Mujika *et al.*, 2017].

Dwell Time in Other Domain Dwell time has been modeled in many domains for a better understanding of user actions. In social media analysis, Yin *et al.* convert dwell time of items to pseudo ratings for prediction of user votes [Yin *et al.*, 2013]. In personalized recommendation, Yi *et al.* directly model normalized dwell time to help estimate the user satisfaction [Yi *et al.*, 2014]. In web searching, Liu *et al.* model the probability of user satisfaction on one click as a product of the click's relevance and the dwell time's information gain [Liu *et al.*, 2017]. However, in the domain of SBP, dwell time is only used to augment the feature vector for user click prediction [Dallmann *et al.*, 2017]. None of the existing methods has jointly predicted user click and dwell time to enhance the performance of recommendation.

3 Dwell time Analysis

3.1 Preliminaries on Survival Analysis

We use capital letters like $X, Y, Z \ldots$ to denote a random variable and their instances are denoted by lower cases $x, y, z \ldots$. Survival analysis aims to model the *survival time* which is the latency until the occurrence of a certain event. Conceptually, dwell time can be viewed as the survival time of leaving the item page and hence are often modeled by survival analysis.

We review some basic concepts and methods in survival analysis [Wang *et al.*, 2017]. Let O being a random variable representing survival time. One way to model O is to suppose the existence of some invertible function $g(\cdot)$ such that

$$g(O) = \mu + \sigma\epsilon, \epsilon \sim f, \tag{1}$$

where f is some simple distribution, e.g. Gaussian and Gamma. The parameters of $g(O)$ can be estimated by maximum likelihood estimation if samples of O is available. However, sampling from an unknown survival model by waiting for the occurrence of the event is unpractical, as the event may not happen after a long observation period. Thus in practice, one predefines a maximum observation duration c and records a tuple (τ, δ) with δ being a boolean value (that takes 0 when the event occurs) and τ being a sample of random variable T

$$T = \begin{cases} O & \text{if } \delta = 0, \\ c & \text{if } \delta = 1. \end{cases} \tag{2}$$

According to [Li *et al.*, 2016], the distribution of $g(T)$ is

$$P(g(T)) \propto \left(f(\frac{g(T) - \mu}{\sigma}) \right)^{1-\delta} \left(1 - F(\frac{g(T) - \mu}{\sigma}) \right)^{\delta}, \tag{3}$$

where F is the Cumulated Density Function (CDF) of f.

3.2 Estimation of Dwell Time

Most e-commerce systems record user clicks along with click timestamps. Let i_k and i_{k+1} be two consecutively clicked items of a session with click time stamps t_k and t_{k+1}. Roughly, the dwell time on i_k is approximated by $\Delta_k = t_{k+1} - t_k$. Such approximation has two drawbacks. Firstly, for the last click of each session, its dwell time is not defined. Secondly, Δ_k may greatly over-estimates the dwell time because a user may be distracted by other activities when viewing the item page. Indeed, in Recsys15 datasets, Δ_k can be as large as 3 hours, which is rarely possible to be the actual dwell time. Moreover, the last dwell time Δ_l cannot be computed from the timestamps as t_{l+1} does not exist. To obtain a more realistic dwell time estimation, we make the following assumptions:

- Users do not lose their attention in c_1 seconds after clicking an item.
- Dwell time of the last click in each session is greater than c_2 seconds.

Here c_1 and c_2 are hyper-parameters. By such assumptions, we obtain a censored dwell time estimation via a tuple

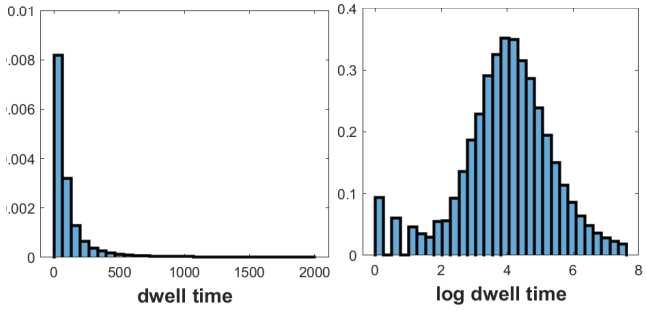

Figure 2: Histograms of dwell time and log dwell time for Recsys15.

(τ_k, δ_k) where $\delta_k = \mathbf{1}\{i_k \text{ is the last click or } \Delta_k \geq c_1\}$ and

$$\tau_k = \begin{cases} \Delta_k, & i_k \text{ is not the last click}, \Delta_k < c_1, \\ c_1, & i_k \text{ is not the last click}, \Delta_k \geq c_1, \\ c_2, & i_k \text{ is the last click.} \end{cases} \quad (4)$$

We refer τ_k as the *censored dwell time*.

3.3 Probability Modeling of Dwell Time

Let D be a random variable representing the dwell time. To motivate our choice in modeling D, we give the histogram of the dwell time of RecSys15 in Fig. 2. This figure shows that the possibility of staying in a single item page decays exponentially fast over the dwell time, and therefore directly modeling D using Gaussian distribution is inappropriate. Surprisingly, the histogram of log dwell time is nearly symmetric around its mode as Fig. 2 shows. Thus, we assume $\log D$ has Gaussian distribution, i.e.

$$\log D = \mu + \sigma\epsilon, \ \epsilon \sim N(0,1) \quad (5)$$

Suppose T is the censored dwell time. According to (3), its density can be formulated as

$$P(\log T) \propto \exp\left(-\frac{(\log T - \mu)^2}{2\sigma^2}\right)^{(1-\delta)} \cdot \left(1 - \Psi(\frac{\log T - \mu}{\sigma})\right)^{\delta} \quad (6)$$

with $\Psi(\cdot)$ the CDF of the standard normal distribution. Note that the density of $\log T$ is not Gaussian.

4 Methodology

Formally, we consider the current session of the form $s_k = \{(i_j, \log \tau_j, \delta_j)\}_{j=1}^{k}$ $(1 \leq k \leq l)$, where i_j is the j-th clicked item, τ_j is the censored dwell time defined in (4), and δ_j is a boolean value indicating whether τ_j is censored or not. Assuming that, the session is sampled from the density $P(S)$ which is factorized as follows.

$$P(S) = \prod_{k=1}^{l} P(I_k, \log T_k | S_{k-1}) \quad (7)$$

$$= \prod_{k=1}^{l} P(I_k|S_{k-1}) P(\log T_k | S_{k-1}, I_k), \quad (8)$$

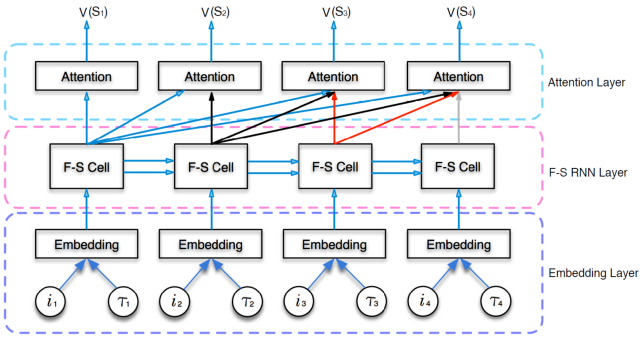

Figure 3: Architecture of JUMP.

where S_{k-1} is the historical clicks and dwell time sequence before the k-th click. $P(I_k|S_{k-1})$ and $P(\log T_k|S_{k-1}, I_k)$ are the conditional probability distributions of the next user click and the dwell time respectively.

The probability distribution of the next user click is assumed to be a soft-max distribution

$$P(I_k|S_{k-1}) \propto \exp(v(S_{k-1})^\top v(I_k)), \quad (9)$$

where $v(S_{k-1})$ is the feature vector of S_{k-1} and $v(I_k) = \mathbf{V}(I_k, :)$ is the embedding vector of item I_k with $\mathbf{V}$ being an embedding matrix and $\mathbf{V}(I_k, :)$ being its I_k-th row. According to (6), the conditional density of log-dwell time is as follows.

$$P(\log T_k | I_k, S_{k-1}) \propto \exp\left(-\frac{(\log T_k - \mu^{(I_k)}(v(S_{k-1})))^2}{2\sigma^2}\right)^{1-\delta_k} \left(1 - \Psi(\frac{\log T_k - \mu^{(I_k)}(v(S_{k-1})}{\sigma})\right)^{\delta_k}, \quad (10)$$

where $\mu^{(I_k)}(v(S_{k-1}))$ is the mean of $\log T_k$. We define $\mu^{(I_k)}(v(S_{k-1})$ as the following linear model[2]

$$\mu^{(I_k)}(v(S_{k-1}) = \mathbf{W}(I_k, :)^\top v(S_{k-1}) + b_{I_k} \quad (11)$$

in which $\mathbf{W}, b$ are parameters of the model. Now the remaining problem is how to extract the feature factor $v(S_{k-1})$ from a session S_{k-1}.

4.1 Extract Session's Feature Vector with RNN

In this section, we will present the RNN that maps a session $S_k (1 \leq k \leq l)$ into a latent vector space. The proposed neural architecture has three layers. From top to bottom, they are referred as the attention layer, the fast-slow layer, and the embedding layer (see Fig. 3).

Attention Layer The attention layer outputs a session's feature vector via performing linear combination

$$v_k = \sum_{i=1}^{l} \alpha(i,k) h_i, \quad (12)$$

[2]Naturally, the function $\mu^{(I_k)}(v(S_{k-1})$ can also be a DNN. However, we found that using a simple linear model is good enough in predicting the dwell time.

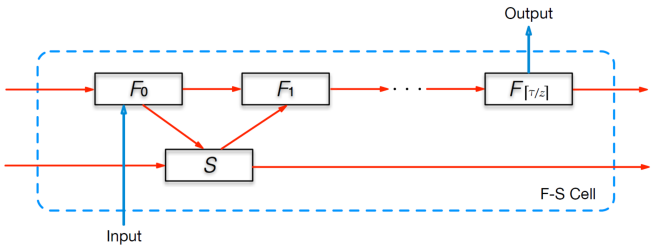

Figure 4: Cell of a Fast-Slow Recurrent Network.

where h_i's are the outputs of the previous fast-slow layer, and the averaging weight $\alpha(i,k)$ is called the attention signal. The attention signal is formulated as

$$\alpha(i,k) = \frac{\exp(h_i^\top h_k/\sqrt{d}))}{\sum_{j=1}^{k}\exp(h_j^\top h_k/\sqrt{d})}. \tag{13}$$

where d is the dimension of the hidden state h_i. Note that $\sum_{i=1}^{k}\alpha(i,k)=1$, and hence the extracted feature vector v_k is equal to a weighted average of the outputs of the fast-slow layer.

Fast-Slow Layer The fast-slow layer is a novel recurrent network. When processing the j-th input $(i_j, \log\tau_j, \delta_j)$, it uses $\lceil\tau_k/s\rceil$ sequentially connected fast cells on the lower hierarchical layer and one slow cell on the top hierarchical layer to model the transition function, as depicted in Fig. 4. The update rule of fast-slow RNN is given as follows.

$$\begin{aligned} h_0^F &= \mathrm{F}(h_{j-1}, x_{(j,\tau_j)}),\\ h_j^S &= \mathrm{S}(h_{j-1}^S, h_1^{F_1}),\\ h_1^F &= \mathrm{F}(h_0^F, h_j^S),\\ h_k^F &= \mathrm{F}(h_{k-1}^F), \text{for } 2\le k\le\lceil\tau_j/z\rceil, \end{aligned} \tag{14}$$

where $F(\cdot,\cdot)$ and $S(\cdot,\cdot)$ are fast unit and slow unit respectively, z is the scale parameter and $x_{(j,\tau_j)}$ is the output of the embedding layer. The output of the j-th step of the fast-slow RNN is $h^F_{\lceil\tau_j/z\rceil}$. Note that the slow cells comprise a sequence which has much longer length than the session. And hence, for a long session, the gradient vanishing/exploding problem would be more severe than GRU.

To attack the above problem, we proposed to use a Recurrent Highway (RH) unit for both slow and fast unit since it has been showed that RH units have better gradient flow than both LSTM and GRU units [Zilly *et al.*, 2016]. Moreover, for capturing the more non-linear relationship between two steps, we use multiplicative information integration [Wu *et al.*, 2016b] which can be used to improve the capacity of RNN cell without increasing model complexity.

Specifically, assuming that h is the previous state and x is the current input, the proposed RH unit updates the state via the following equations:

$$\begin{aligned} m &= \alpha\odot(\mathbf{W}_c x\odot\mathbf{U}_c h)+\beta\odot\mathbf{W}_c x+\gamma\odot\mathbf{U}_c h,\\ c &= \tanh(m),\\ g &= \sigma(\mathbf{W}_g x+\mathbf{U}_c h),\\ h' &= (1-g)\odot h+g\odot c, \end{aligned} \tag{15}$$

Table 1: Basic Statistics of the Benchmark

dataset	RECSYS15	CIKM16	REDDIT
#items	37,961	35,490	1,6430
#clicks	31,710,143	593,708	3,304,642
#train sessions	6,385,771	157,038	808,489
#test sessions	1,596,442	39,259	202,123
avg. length	3.972	3.024	3.264
max dwell time	6,200	3,900	3,600

where $\odot$ is the point-wise product, α is multiplicative gate unit, β,γ are linear gate unit, and h' is the output of RH unit. For units without input, all the terms containing x will be omitted.

Embedding Layer The bottom layer is the embedding layer which maps a tuple $(i_k, \log\tau_k)$ to a vector. We jointly embed dwell time since it have been found that such approach is is beneficial for enhancing the prediction accuracy [Dallmann *et al.*, 2017]. Specifically, the embedding is done by batch-normalizing

$$x_{i_k,t_k} = \mathrm{BatchNormalize}(\mathbf{V}(i_k,:)+\mathbf{T}(\lceil\tau_k\rceil,:)), \tag{16}$$

where $\mathbf{V}$ is the item embedding matrix and $\mathbf{T}$ is the time embedding matrix. Note that, in the considered datasets, the dwell time of users are less than 6000 seconds. Thus, the row number of the embedding matrix $\mathbf{T}$ is less than 6000.

4.2 Composition of Likelihood

Given dataset of N sessions $\mathbf{D}=\{s^{(n)}\}_{n=1}^{N}$ with each session being sequences $s^{(n)}=\{(i_k^{(n)},\log\tau_k^{(n)},\delta_k^{(n)})\}_{k=1}^{l^{(n)}}$, the log-likelihood of the dataset is

$$L(\theta)=\frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{l^{(n)}}(\log P(i_k^{(n)}|s_{k-1}^{(n)}) \tag{17}$$

$$+\log P(\log\tau_k^{(n)}|i_k^{(n)},s_{k-1}^{(n)})), \tag{18}$$

where θ is the parameters of JUMP and the conditional densities $P(i_k^{(n)}|s_{k-1}^{(n)})$ and $P(\log\tau_k^{(n)}|i_k^{(n)},s_{k-1}^{(n)}))$ are defined in (9) and (10). The parameter θ can be trained by maximizing the log-likelihood.

5 Experiments

To valid the performance of our model, we perform the task of user click prediction and dwell time estimation. State-of-the-art session-based recommendation methods, which include GRU [Hidasi *et al.*, 2015], IGRU [Hidasi and Karatzoglou, 2017], NARM [Li *et al.*, 2017], DTGRU [Li *et al.*, 2016], are compared. We also compare ours with neural temporal process models including RMTP [Du *et al.*, 2016], A-TRP [Xiao *et al.*, 2017], NSR [Jing and Smola, 2017]. All the compared methods are performed on the same PC with i7-7820HK CPU, 16GB RAM, and GTX1080 GPU. All the compared methods are optimized by Adam with the batch size set to 100.

5.1 Datasets

We conduct experiments on three publicly available datasets including RecSys15, CIKM16, and REDDIT. RecSys15 is used for RecSys Challenge 2015. It contains click streams with timestamps collected from a commerce site. CIKM16 is from CIKMCup 2016. It contains sequences of anonymous transactions provide by DIGINETICA. REDDIT contains a log of user interactions on different subreddits (sub-forums) at different timestamps. The original REDDIT dataset does not organize user interactions into sessions. To re-organize the dataset, we group the consecutive actions that happened within the time of one hour into a session. For all the datasets, items or subreddits that appear less than 3 times are discarded, and the sessions with length equal to one are filtered out. After such process, we split all sessions of the datasets into 80% for training and 20% for testing. Note that each session is assigned to either training or testing. We list the basic statistics of the datasets in Tab. 1.

5.2 Experiments on Item Prediction

To compare the top-k recommendation performance of our method with the baselines, we evaluate recall@20, MRR@20 and NDCG@20 of these methods on RECSYS15, CIKM16, and REDDIT datasets. For all the methods in our comparison, the dimension of item embedding vectors is set to 100 to make the number of their free parameters to be the same order. Besides, the dimension of the hidden states in the recurrent units are chosen to be the same with the word dimension for simplicity. For our model, we set σ to 10, $c1 = 2000$ and $c2 = 30$. $c1$ and $c2$ is selected from a candidate set $20, 30, 200, 1000, 2000, 20000$ which lead to best prediction accuracy on randomly sampled 10% data from Recsys15. The results are showed in Tab. 3. From Tab. 3 we can see that JUMP outperforms all the baselines on all the datasets. This can be explained by that our model integrates the dwell time more effectively than the baselines and that our model uses a more refined RNN structure.

To verify the performance of these methods under different session lengths, we evaluate recall@20 for short sessions (length from 2 to 6), moderate sessions (length from 10-30), and long sessions (length greater than 150). We report the results in Fig. 7. From the figure, we can see that all the methods perform nearly equally well at modest length sessions while for short and long sessions, the performance of all the methods deteriorate to some extent. For short sessions, our method outperforms NARM and IGRU significantly, and this is because our model takes dwell time information in neural network designation. For long sessions, the performance of JUMP and NARM are similar, and both outperform IGRU, this is because JUMP and NARM use the attention mechanism to stabilize the performance of long step RNN. To check the performance of JUMP when different state dimensions are used (the item dimension is set to the same value as the state dimension), we compare it with NARM and IGRU under different dimension settings in Fig.6. Since dimension d controls the expressive power of the models, it has a huge impact on the performance. The results show that our method has better click prediction accuracy than NARM and IGRU under all the dimension settings.

Table 2: Performance of Compared Models on Time Prediction Task

	RECSYS15	CIKM16	REDDIT
method	MAE	MAE	MAE
ATRP	95.23	229.84	271.8
RMTP	289.0	290.7	302.4
NSR	283.0	286.2	225.1
JUMP	**73.61**	**180.6**	**185.7**

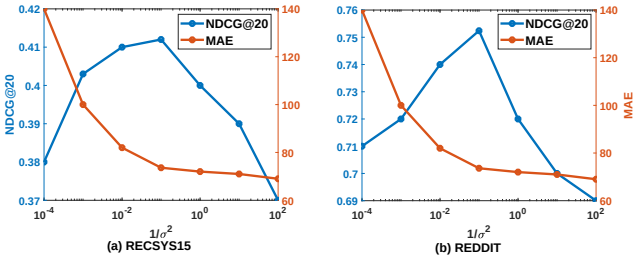

Figure 5: Trade off between click prediction and dwell time prediction

5.3 Experiments on dwell Time Prediction

We validate the accuracy of dwell time modeling of JUMP via comparing the performance with neural temporal process models including RMTP, ATRP, and NSR. The accuracy of the time prediction is measured by Mean Absolute Error (MAE). The time prediction accuracy results are reported in Tab. 2. We can see that our model has much smaller MAE than all the other models. We attribute the performance gain to the ability of our model to utilize user clicks of the SBP datasets. The hyperparameter σ controls the trade-off between the accuracy in predicting dwell time and the accuracy of click forecasting. To depict this, we vary $1/\sigma^2$ from 10^{-4} to 100 and report the prediction accuracies of next click and dwell time in Fig. 5. From the figure, we can see that, when $1/\sigma^2$ is relatively small, both the accuracy of click prediction and time prediction grows as $1/\sigma^2$ increases. In this phase, dwell time prediction task and click prediction task cooperate with each other to find a better representation for the sessions. When $1/\sigma^2$ grows relatively large, time prediction accuracy still grows while click prediction accuracy decays from the peak. In this phase, the model treats the time prediction as a more important task and compromises the click prediction accuracy for the time prediction performance.

6 conclusion

We explore using survival analysis and sequential modeling techniques to jointly predict user clicks and dwell time for the session-based recommendation. We propose a new predictor, JUMP. JUMP contains a new three-layered neuron network to extract a feature vector from input session. The network utilizes attention mechanism to improve the robustness against noise. And the fast-slow RNN structure is embedded to boost performance for very short sessions. Empirical results validate the performance of the proposed method.

Table 3: Performance of Compared Models

	RECSYS15			CIKM16			REDDIT		
method	Recall(@20)	MRR	NDCG	Recall(@20)	MRR	NDCG	Recall(@20)	MRR	NDCG
GRU	0.6020	0.2473	0.3267	0.4728	0.1258	0.2020	0.7433	0.5988	0.6144
IGRU	0.6905	0.2955	0.3843	0.5467	0.1502	0.2371	0.8257	0.7011	0.7355
NARM	0.6907	0.2969	0.3845	0.5694	0.1902	0.28530	0.8034	0.6955	0.7122
DTGU	0.6577	0.3015	0.3899	0.3755	0.1344	0.1877	0.7833	0.6544	0.6984
ATRP	0.6717	0.2495	0.3275	0.3666	0.1331	0.1966	0.8131	0.6406	0.6811
RMTP	0.6716	0.3035	0.3941	0.5674	0.1786	0.2603	0.8143	0.7037	0.7198
NSR	0.6691	0.2835	0.3715	0.5235	0.1554	0.2365	0.8235	0.7029	0.7311
JUMP	**0.7168**	**0.3238**	**0.4120**	**0.6046**	**0.2080**	**0.3049**	**0.8434**	**0.7241**	**0.7524**

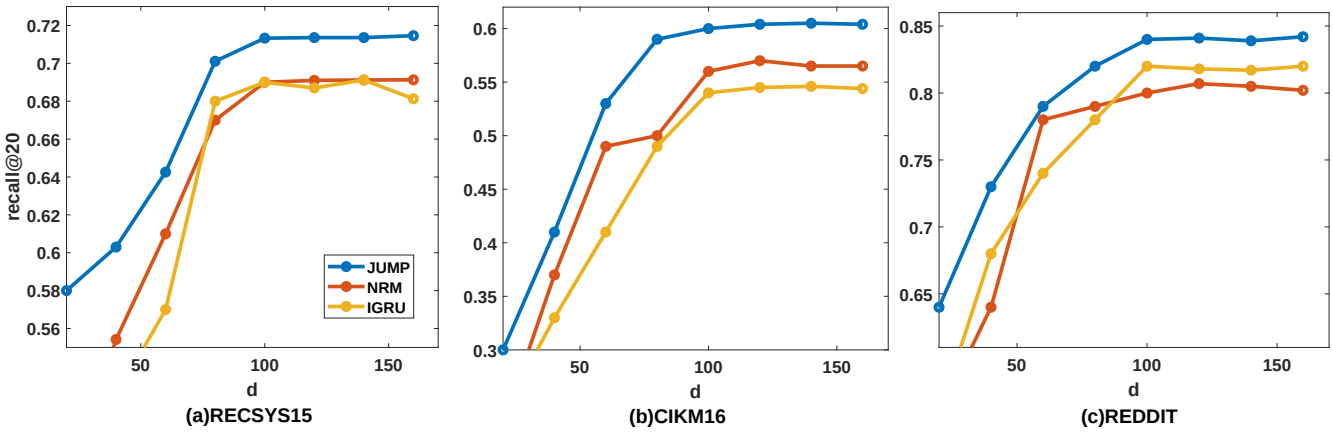

Figure 6: Performance of compared methods under different dimensions

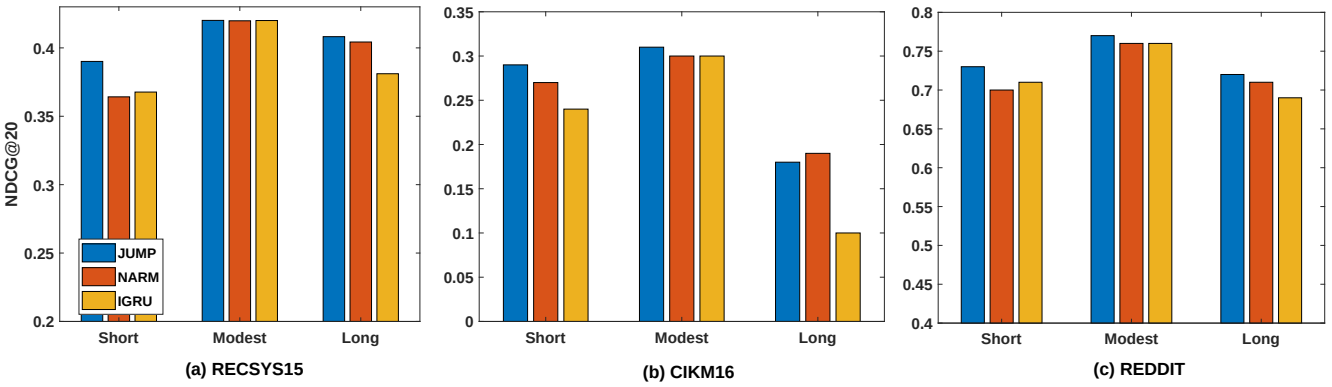

Figure 7: Performances of compared methods with respect to short sessions, modest length sessions, and long sessions

Acknowledgments

This work is supported by National Natural Science Foundation of China (Grant No: 61472347, 61672376, 61751209), and Zhejiang Provincial Natural Science Foundation of China under Grant No. LZ18F020002.

References

[Bahdanau *et al.*, 2014] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv*, 2014.

[Baxter, 1997] Jonathan Baxter. A bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine learning*, 28(1):7–39, 1997.

[Bengio *et al.*, 1994] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[Chung *et al.*, 2014] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv*, 2014.

[Dallmann *et al.*, 2017] Alexander Dallmann, Alexander Grimm, Christian Pölitz, Daniel Zoller, and Andreas Hotho. Improving session recommendation with recurrent neural networks by exploiting dwell time. *arXiv*, 2017.

[Du *et al.*, 2016] Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. Recurrent marked temporal point processes: Embedding event history to vector. In *SIGKDD*, pages 1555–1564. ACM, 2016.

[Hidasi and Karatzoglou, 2017] Balázs Hidasi and Alexandros Karatzoglou. Recurrent neural networks with top-k gains for session-based recommendations. *arXiv*, 2017.

[Hidasi and Tikk, 2016] Balázs Hidasi and Domonkos Tikk. General factorization framework for context-aware recommendations. *Data Mining and Knowledge Discovery*, 30(2):342–371, 2016.

[Hidasi *et al.*, 2015] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *arXiv*, 2015.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[Jing and Smola, 2017] How Jing and Alexander J Smola. Neural survival recommender. In *WSDM*, pages 515–524. ACM, 2017.

[Li *et al.*, 2016] Yan Li, Kevin S Xu, and Chandan K Reddy. Regularized parametric regression for high-dimensional survival analysis. In *SDM*, pages 765–773. SIAM, 2016.

[Li *et al.*, 2017] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. Neural attentive session-based recommendation. In *CIKM*, pages 1419–1428. ACM, 2017.

[Lin *et al.*, 2017] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. *ICLR*, 2017.

[Linden *et al.*, 2003] Greg Linden, Brent Smith, and Jeremy York. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, 7(1):76–80, 2003.

[Liu *et al.*, 2017] Yiqun Liu, Xiaohui Xie, Chao Wang, Jian-Yun Nie, Min Zhang, and Shaoping Ma. Time-aware click model. *ACM Transactions on Information Systems*, 35(3):16, 2017.

[Mujika *et al.*, 2017] Asier Mujika, Florian Meier, and Angelika Steger. Fast-slow recurrent neural networks. In *NIPS*, pages 5917–5926, 2017.

[Musto *et al.*, 2015] Cataldo Musto, Giovanni Semeraro, Marco De Gemmis, and Pasquale Lops. Word embedding techniques for content-based recommender systems: An empirical evaluation. In *RecSys*, 2015.

[Quadrana *et al.*, 2017] Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. Personalizing session-based recommendations with hierarchical recurrent neural networks. In *RecSys*, pages 130–137. ACM, 2017.

[Ruder, 2017] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv*, 2017.

[Ruocco *et al.*, 2017] Massimiliano Ruocco, Ole Steinar Lillestøl Skrede, and Helge Langseth. Inter-session modeling for session-based recommendation. In *Proceedings of the 2nd Workshop on Deep Learning for Recommender Systems*, pages 24–31. ACM, 2017.

[Shani *et al.*, 2005] Guy Shani, David Heckerman, and Ronen I Brafman. An mdp-based recommender system. *JMLR*, 6(Sep):1265–1295, 2005.

[Suarez, 2017] Joseph Suarez. Language modeling with recurrent highway hypernetworks. In *NIPS*, pages 3269–3278, 2017.

[Sukhbaatar *et al.*, 2015] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In *NIPS*, pages 2440–2448, 2015.

[Tavakol and Brefeld, 2014] Maryam Tavakol and Ulf Brefeld. Factored mdps for detecting topics of user sessions. In *RecSys*, pages 33–40. ACM, 2014.

[Wang *et al.*, 2017] Ping Wang, Yan Li, and Chandan K Reddy. Machine learning for survival analysis: A survey. *arXiv*, 2017.

[Wu *et al.*, 2016a] Sai Wu, Weichao Ren, Chengchao Yu, Gang Chen, Dongxiang Zhang, and Jingbo Zhu. Personal recommendation using deep recurrent neural networks in netease. In *ICDE*, pages 1218–1229. IEEE, 2016.

[Wu *et al.*, 2016b] Yuhuai Wu, Saizheng Zhang, Ying Zhang, Yoshua Bengio, and Ruslan R Salakhutdinov. On multiplicative integration with recurrent neural networks. In *NIPS*, pages 2856–2864, 2016.

[Xiao *et al.*, 2017] Shuai Xiao, Junchi Yan, Mehrdad Farajtabar, Le Song, Xiaokang Yang, and Hongyuan Zha. Joint modeling of event sequence and time series with attentional twin recurrent neural networks. *arXiv*, 2017.

[Yi *et al.*, 2014] Xing Yi, Liangjie Hong, Erheng Zhong, Nanthan Nan Liu, and Suju Rajan. Beyond clicks: dwell time for personalization. In *RecSys*, pages 113–120. ACM, 2014.

[Yin *et al.*, 2013] Peifeng Yin, Ping Luo, Wang-Chien Lee, and Min Wang. Silence is also evidence: interpreting dwell time for recommendation from psychological perspective. In *KDD*, pages 989–997. ACM, 2013.

[Zhang *et al.*, 2014] Yuyu Zhang, Hanjun Dai, Chang Xu, Jun Feng, Taifeng Wang, Jiang Bian, Bin Wang, and Tie-Yan Liu. Sequential click prediction for sponsored search with recurrent neural networks. In *AAAI*, volume 14, pages 1369–1375, 2014.

[Zilly *et al.*, 2016] Julian Georg Zilly, Rupesh Kumar Srivastava, Jan Koutník, and Jürgen Schmidhuber. Recurrent highway networks. *arXiv*, 2016.