School of Informatics

Informatics Project Proposal
Timing Aware Multi-day Window Recommender

B231968
April 2023

Abstract

Sequential recommender systems widely employ a next item/action prediction approach based on a user's behavior trajectories. Recently the model that considers time intervals between items has shown promising performance. The latest model employs a multi-day window approach predicts future actions, but so far, evaluation has only been on a single, private dataset. We evaluate the latest approach to public datasets by extending a next-item predictor. We explore two time-encoding schemes for 1. a multi-item predictor using the explicit time interval between every item to compare with a next-item predictor and 2. a multi-day items predictor using the implicit time interval between input and target times to compare with our proposed model. Building on those, we propose a Timing Aware Multi-day Window Self-Attention Recommender that learns "when and what" to recommend and predicts a sequence of multi-day items and their highest close rate day pairs.

Date: Wednesday 26th April, 2023

Tutor: Michael Herrmann
Supervisor: Michael la Grange, David Wardrope, Timos Korres

1 Motivation

Recommender systems cover a wide range of commercial uses and research topics in personalized machine learning where quality human recommendations or big data analysis are unfeasible, such as predictions on e-commerce products, e-movies, internet games, and online user's actions. Sequential recommender systems are their branch and widely employ a next-action prediction approach. The model predicts a user's next action based on their behavior history, and various algorithms like Markov Chains (MCs), Recurrent Neural Networks (RNNs), and Self-Attention (SA) have been used to model sequential patterns. Recent studies have shown that accounting for time intervals between items improves the performance of next-item prediction. The latest study discussed that future-item prediction by a multi-day window predictor is more performant than next-item prediction [1]. However, the effectiveness of the approach and its generalized performance are unclear, and the impact of time feature encoding schemes on those predictors have not been explored. In our work, we will extend and improve the existing model, and evaluate the multi-day window prediction approach on diverse public datasets. We explore two time-encoding schemes for 1. a multi-items predictor using the explicit time interval between every item used in [2] to compare with a next-item predictor and 2. a multi-day items predictor using the implicit time interval between input and target times used in [1] to compare with our proposed model. Building on those, we propose a Timing Aware Multi-day Window Self-Attention Recommender that learns "when and what" to recommend and predicts a sequence of pairs of a purchased item and its highest close rate day.

1.1 Problem Statement

The problem with previous sequential recommenders is that they assume predicting their single next purchase is the best learning task for recommender systems. The range that influences the prediction is only up to the next item based on the previous ones. However, intuitively, recommendation strategy and item purchase frequency will have more influence on future recommendations. For instance, two sales representatives have the same customer's cosmetic purchase history. Seller 1 recommends a face cream next time two weeks later because the customer purchased it several times. On the other hand, seller 2 recommends a lip cream next time and then the face cream after that because she knows the customer wouldn't buy the face cream as it lasts one month but may repurchase the lip cream purchased three months ago, i.e., she knows "when and what" to recommend. Therefore, the strategies of the two sellers and item purchase frequency should have different impacts on the future purchases, even if they have the same purchase history. There is only Seller 1's strategy is available in existing next-item predictors, and we need to solve this problem by modeling Seller 2's strategy, filling the gap between human and AI recommendations.

1.2 Research Hypothesis and Objectives

Based on the problem statement, we hypothesize that a multi-day window predictor could improve performance by extending the window length depending on a dataset because it considers the user's interest in the long-term timeframe, and purchase frequency is different by item and user. Previous work disregarded them and its strategy might be improved, and it might not have reflected the item purchase frequencies in their prediction. Besides, the influence of the time encoding scheme is unknown. We propose a timing aware multi-day window self-attention mechanism to address these limitations. The model predicts a sequence of multi-day items and

their highest close rate day pairs that a user will be interested in over the next K items with the flexibility of window length while considering the time intervals between every item.

Our primary objective is to learn a model that predicts a user's future purchases over multi-day windows rather than a traditional next-item prediction. We choose the window length depending on the dataset, as the average number of sequence lengths varies. We assume that items a user is interested in within the window represent a user's longer-term interests sufficiently. Figure 2 in Appendix illustrates our model framework, and we expand on significant components in more detail in the Methodology section. The measurable objective is the performance improvement ($ndcg@k$ and $hit@k$) from TiSASRec [2] in item prediction (see Evaluation) on MovieLens-1m, Amazon Beauty, and Amazon Game datasets. Our study will focus on sequential item prediction. We will not address action prediction (e.g., click, scroll, hide, etc.) or language prediction (e.g., text generation, etc.) though one can expand our model to those purposes.

1.3 Timeliness and Novelty

We conduct timely research that extends TiSASRec (2020) [2], a recent sequential recommendation model that considers time intervals between items and shows promising results, based on the latest theory by PinnerFormer (2022) [1] of predicting a sequence of items. We identified that TiSASRec is limited to next-item prediction, and no previous study evaluated Pinnerformer's theory for general use. We aim to extend the model for public datasets and investigate the timestamp encoding scheme. Besides, our model predicts both "when and what" to recommend, which has not been achieved by any previous model. Our research offers a new paradigm in the personalized sequential recommendation, making it highly novel.

1.4 Significance

We suggest structural changes to implement a latent factor model for a multi-day window predictor in a next-item prediction model [3, 2]. This model lacks day-factor or time-factor in its predicted item, i.e., predict Item_{t+b} means $t + b$ th item, not the item on day $t + b$, making it challenging to extend the window length for day $t + b$ prediction. We propose exploring timing-aware multi-day window prediction using the Transformer, a default choice for many sequential recommendation problems, and no existing way to do this has been proposed. Common domains for future item recommendations include e-commerce, Point-of-Interest, music, and movies. Future work could experiment with more time modeling to improve performance, especially for datasets with limited purchase history. Time modeling could predict the timing of recommendations based on a few purchased items, narrowing down customer preferences and enabling recommendations even with limited purchase history.

1.5 Feasibility

The proposed pipeline for timing aware multi-day window item recommendation involves timestamp encoding and multi-item training. We have identified alternative machine learning models in related work and methodology, such as Time2vec [4] and TiSASRec [2], with publicly available code implementations that we can use for our pipeline instead of building and training a model from scratch. We will explore different heuristics in training and tuning processes to produce multiple models for our experiments, depending on our progress to ensure we can complete the project within the 10-week time frame.

1.6 Beneficiaries

The proposed work benefits users of the system and the research field of sequential recommendation and personalized machine learning. It serves as a tool for researchers, developers, and e-commerce marketers interested in predicting future trends and providing users recommendations with more control. The system's recommendations may lead to commercial profits. We address the limitations of existing models and propose two ways to predict future items as Model1 and Model2 (see Methodology), providing a benchmark for other researchers and inspiring further innovations in future item recommendation.

2 Background and Related Work

2.1 Sequential Recommendation

Sequential recommendation systems are machine learning systems that take a sequence of user's behavior trajectories, e.g., a sequence of purchased item IDs, as input and adopt recommendation algorithms to output a next item the user would be interested in by providing a ranked item list. We choose a data split strategy among options (e.g., leave one last item, temporal split, and random split, etc. [5]) by purpose for training and testing the recommendation model. For instance, the leave one last item strategy uses the final transaction per user for testing, the second last one for validation, and the remaining transactions for training. Classical sequential recommendation approaches, such as frequent pattern mining, K-nearest neighbors, Markov chains (MCs), and reinforcement learning, typically adopt matrix factorization and tend to fail capturing users' long-term patterns by combining different sessions [6]. In general, MC-based methods perform best in learning some notion of context based on the user's recent actions with extremely sparse datasets, where model parsimony is critical [3]. Given the success of applying sequence modeling to various natural language processing (NLP) tasks, Deep Learning (DL) based approaches, such as Recurrent neural networks (RNNs) and Convolutional neural networks (CNNs), have been applied to sequential recommendation modeling. Specifically, RNNs capture the dynamics of sequential patterns and dependencies well among items in item-based recommendation tasks, and perform much better recommendations over the classical models [6]. RNNs learn users' long-term semantics well with denser datasets, where higher model complexity is affordable.

2.2 Attention Mechanisms

We can intuitively understand attention mechanisms in DL as human visual attention (the tendency to be attracted to more significant parts of an object). That is, the output depends on certain traits of the relevant input. Such a mechanism can compute input weights and make the model more interpretable. It was initially proposed by Bahdanau et al. [7] in a neural machine translation task, focusing on modeling the importance of different parts of an input sentence to output words. Based on this research, it has also been widely applied in NLP and computer vision, where it shows to be effective in various tasks such as machine translation [8] and image captioning [9]. In sequential recommendations, vanilla attention is proposed and widely used by applying for this work as a decoder for RNNs [10].

2.3 Self-Attention Mechanisms

Self-attention mechanisms, initially developed by Google as Transformer in 2017 [11], have gained interest in sequential recommendations. They estimate the weight of each item in a user's interaction trajectory to learn their short-term intentions and long-term interests. Compared to RNN-based models, self-attention performs better in sequential recommendations [12]. In SDM [13], a multi-head self-attention module captures a user's short-term preferences, while attention and dense-fully-connected networks encode their long-term interests. SASRec [3] employs self-attention to predict the next item from a user's purchase history. Since the self-attention model is not aware of the positions of previous items, [14, 11] add positional encodings to the inputs, learning a deterministic function or positional embedding. Alternatively, [7] model the relative positions between two input elements as pairwise relationships. TiSASRec [2] combines absolute and relative positions to design time-aware self-attention, which improves SASRec by modeling items' positions and time intervals. Pinnerformer [1] attempts to predict actions multiple days ahead rather than predict the next item, capturing the long-term interest of users. Our model is inspired by Pinnerformer and uses TiSASRec as a baseline to predict multiple items ahead while considering their highest close rate dates, aiming to explore the latest method by [1] further and improve model performance.

3 Programme and Methodology

3.1 Time-Interval Aware Multi-item Window Self-Attention: Model1

As a first step of the project, we implement a Time-Interval Aware Multi-item Window Self-Attention Recommender, that effectively captures personalized time and interest pattern to recommend the most relevant multi items, i.e., the model knows "what" to recommend in "which" order. Going beyond the state-of-the-art, the goal of this model is to 1. provide a ranked item list and establish its performance as a multi-item window predictor and 2. explore influence of time feature encoding scheme and loss function on the item prediction, which is an unexplored area. To achieve the goal, we manage the project by making three modifications on our baseline [2]: (a) add two time encoding options, (b) extend the baseline into a multi-item window predictor, and (c) adopt two loss functions.

3.2 (a) Time Feature Encoding

We explore two encoding schemes for capturing time to learn the user's longer-term interests. Each matrix is used to get embedding matrices to compute loss.

Option 1: Time Interval as a Square Relation Matrix.

This is used to predict the i-th purchase item from the input item. To represent the personalized time interval, we adopt [2] and encode the relative length of time intervals in one user sequence as a relation matrix. This captures the user's interaction frequency, e,g., some users have more frequent interactions while others do not. For all time intervals on n items, a time interval of two items i and j, $|ti - tj|$, is divided by the minimum time interval $r^u_{min} = min(\boldsymbol{R}^u)$ (other than 0) of a user, creating the scaled time intervals $r^u_{ij} = [\frac{|t_i - t_j|}{r^u_{min}}]$, resulting $\boldsymbol{M}^u \in N^{n \times n}$ of user u.

$$\boldsymbol{M}^u = \begin{bmatrix} r^u_{11} & r^u_{12} & \dots & r^u_{1n} \\ r^u_{21} & r^u_{22} & \dots & r^u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ r^u_{n1} & r^u_{n2} & \dots & r^u_{nn} \end{bmatrix}$$

Option 2: Time Interval as a Matrix of Action Vectors. This is used to predict the purchased item on i-th day from input time and to reproduce PinnerFormer [1]. To represent the time an action occurred, we adapt [1] and encode 1. raw absolute timestamp, 2. time since the latest action a user has taken, and 3. time gap between actions. We follow the same procedure as [1] and create $2P + 1$ features per action using sine and cosine transformations with various periods in a manner similar to Time2vec [4], but with fixed periods, rather than learned periods, and a logarithmic transformation of time, rather than a linear one, where P is the number of periods set manually.

This ensures the model can distinguish between short durations (e.g., 10 seconds vs. 1 minute), as compared to long durations (e.g., 10 days vs. 11 days). All features are then concatenated into a single vector, resulting in an input vector of dimension D_{in}. This represents an action of the user $S_i^u \in \mathbb{R}^{D_{in}}$, where $S_t^u \in I$ (item set), and is used to form an input matrix $\boldsymbol{A}^u \in \mathbb{R}^{M \times D_{in}}$ of user u, using M actions.

$$\boldsymbol{A}^u = \begin{bmatrix} a^u_{T1} & a^u_{T2} & \cdots & a^u_{TD_{in}} \\ a^u_{(T-1)1} & a^u_{(T-1)2} & \cdots & a^u_{(T-1)D_{in}} \\ \vdots & \vdots & \vdots & \vdots \\ a^u_{(T-M+1)1} & a^u_{(T-M+1)2} & \cdots & a^u_{(T-M+1)D_{in}} \end{bmatrix}$$

3.3 (b) Multi-item Window Predictor

To extend the baseline, we define the problem formulation as follows. Let U and I represent the user and a set of items, respectively. In the setting of time-aware sequential recommendation, for each user $u \in U$, we are given the user's action sequence, denoted as $S^u = (S_1^u, S_2^u, ..., S_{|S^u|}^u)$ where $S_t^u \in I$ and time sequence $T^u = (T_1^u, T_2^u, ..., T_{|T^u|}^u)$ corresponding to the action sequence. During the training process, at time step t, the model predicts a sequence of items, e.g., $(\text{Item}_{t+3}, \text{Item}_{t+12}, \text{Item}_{t+21})$, based on the previous t items and the time intervals r_{ij}^u between item i and j (input time i and target time j if encoding option 2). Our model's input is $(S_1^u, S_2^u, ..., S_{|S^u|-1}^u)$ and time intervals R^u in the sequence, where $R^u \in N^{(|S^u|-1)\times(|S^u|-1)}$. The desired output is a sequence of the next items at the latest input time S_t^u (Figure 2 in Appendix).

Prediction Layer. We follow the procedure in TiSASRec and get $\boldsymbol{Z}_t$, the combined representation of items, positions, and time intervals, from their Time Interval-Aware Self-attention Layer [2], and create $\boldsymbol{M}^I$, the item embedding matrix. To predict the Item_{t+b} where b is the target item, we compute users' preference score of item i at this prediction layer: $A_{i,t} = \boldsymbol{Z}_t \boldsymbol{M}_i^I$, where $\boldsymbol{M}_i^I \in \mathbb{R}^d$ is the embedding of item i and $\boldsymbol{Z}_t$ is the representation given the first t items (i.e., $s_1, s_2, ..., s_t$) after preprocessing and their time intervals (i.e.,$r_{1(t+b)}^u, r_{2(t+b)}^u, ..., r_{t(t+b)}^u$) between the $(t + b)$th item. For instance, to predict Item_{t+7}, b in $r_{t(t+b)}^u$ is replaced with 7.

3.4 (c) Loss Function

We adopt two loss functions to test on our model. Since user interactions are implicit data, we cannot directly optimize $A_{i,t}$. Hence, we adopt negative sampling to optimize the ranking of items with the Adam optimizer [15], applying mini-batch SGD. We define $o = (o_1, ..., o_n)$ as the expected output given a time and item sequences, $t = (t_1, ..., t_n)$ and $s = (s_1, ..., s_n)$, respectively. For instanc, $o_1 = s_{t+7}$ if $1 \leq t + 7 < n$, describing the output for the item at $t + 7$.

Binary cross entropy. For each expected positive output o_i, we sample a negative item

$o'_i \notin S^u$ to generate a set of pairwise preference orders $D = (S^u, T^u, o, o')$ as a training sample. We transform preference scores into the range (0, 1) by a sigmoid function $\sigma(x) = 1/(1 + e^{-x})$ and adopt binary cross entropy as the loss function:

$$L = -\sum_{S^u \in S} \sum_{t \in [1,2,...,n]} [\log(\sigma(a_{o_t,t})) + \log(1 - \sigma(a_{o'_t,t}))] + \lambda \|\Theta\|_F^2 \tag{1}$$

where $a_{o_t,t}$ is the preference score of an output item o_t given the first t items, Θ is the set of embedding matrices created by following procedure in TiSASRec [2], $\|\cdot\|_F$ denotes the Frobenius norm, and λ is the regularization parameter. Note that we mask the loss of the padding item.

Sampled Softmax Loss Function. In the original paper as our baseline, the TISASRec model is trained based on a binary cross-entropy task using the above loss function, without any sample probability correction for a next item prediction. Inspired by [1], we try adopting their sampled softmax with a $\log Q$ correction [16] in our experiments, only allowing positive preference scores to contribute to the model's loss. For each expected positive output o_i, we produce a set of negative samples $(o'_1, ..., o'_n)$. We compute a loss for each output and then compute a weighted average such that each user in the batch is given equal weight. Our sampled softmax loss with a $\log Q$ correction is defined as:

$$L(u, o_t) = -\log\left(\frac{e^{\log(\sigma(a_{o_t,t})) - \log(Q(o_t))}}{e^{\log(\sigma(a_{o_t,t})) - \log(Q(o_t))} + \sum_{j=1}^{N} e^{\log(1-\sigma(a_{o'_t,t})) - \log(Q(o'_j))}}\right) \tag{2}$$

where $Q(v) = \mathrm{P}(\text{Item } v \text{ in batch} \mid \text{User } u \text{ in batch})$ is a correction term, the sampling probability of item v in a random batch.

3.5 Training Objectives.

We adopt three training objectives from [2, 1] that are described below, and are depicted in Figure 3 in Appendix, aiming to achieve using the user's action sequence $(S^u_1, S^u_2, ..., S^u_{|S^u|-1})$.

Next Item Prediction. Predict a next item Item_{t+1}.

All Item Prediction. Predict all items a user will be interested in over the next x-items, e.g., (Item_{t+3}, Item_{t+8}, and Item_{t+12}), using the final user embedding e_t, all of which fall within a x-item window of t. This objective forces the model to learn longer-term interests.

Dense All Action Prediction. Predict all items a user will be interested in over the next x-items using a set of random indices, s_i, and for each e_{s_i}, aim to predict a randomly selected item from the set of all items over the next x items. e_t is used as our final user representation in inference.

3.6 Timing-Aware Multi-day Window Self-Attention: Model2

The limitation of the first step is the model doesn't recommend items in a timely manner as a human sales representative does. Similarly, PinnerFormer predicts the i-th day action, but in the case of item prediction, it's not necessarily the day with a high closing rate, and we cannot assume positive engagement. As a second step of the project, we propose a Timing-Aware Multi-day Window Self-Attention Recommender that effectively captures personalized time and interest patterns to learn "when and what" to recommend and predicts the most relevant item on the day with the highest closing rate. This version not only predicts the item but also the time that is likely to be purchased, i.e., user's future positive engagement day. To

achieve this objective, in addition to the item predictor, we add another model called the time predictor using the same training examples as input. For instance, the model outputs a pair of (Item_{t+b}, date), where we calculate the date from the predicted time interval relative to the time of an input event, e.g., (Item_{t+1}, +8 day).

3.7 Risk Assessment

Table 1 lists the possible risks of our project, which we describe in order. The highest risk, implementation trouble, could prevent us from making progress on the project. However, we can minimize this risk by ensuring the reproduction of the baseline code before the project starts. The second risk is that our extended model may not work initially due to the length of the user's action sequence, but we can experiment with different datasets or window sizes to establish model performance. The third risk is struggling with encoding time features, but we can study Time2vec's published code [4] at the early stage of the project to implement the encoding scheme. The fourth and final risk is that the sampling softmax loss function we plan to adopt from [1] may not incompatible with our model, in which case we can use the baseline's binary cross-entropy loss function or search for an alternative. It may make sense to use the baseline scheme for risks 3 and 4 and use the time for the Model2 rather than getting stuck if it takes much longer to make it work.

	Risk name	Severity	Likelihood	Mitigation
1	Implementation takes longer	High	Low	Reproduce baseline by May
2	Model incompatible with chosen dataset or window size	Medium	Medium	Use other dataset / reduce window size
3	Time encoding takes longer	Medium	Medium	Reivew Time2vec / pick baseline scheme
4	Softmax loss not working	Low	Medium	Pick binary-cross entropy or alternative

Table 1: Risk assessment of the project

3.8 Ethics

Our experiments with public datasets do not raise any ethical concerns, and the code would not directly harm researchers or corporate developers who use it. However, our model's recommendations may lead to biased information being presented to end-users, depending on the datasets used and the specific use case. For example, if a woman's purchase history on an e-commerce website consists only of shoes, the model may only recommend shoes and not consider her potential interest in other products such as cosmetics. Firms integrating our model should take these biases into consideration. We will comply with the General Data Protection Regulation and the policy of the University of Edinburgh by following the School of Informatics ethics procedure and submitting an Informatics Ethics Form for review.

4 Evaluation

We evaluate our model in aspects of two predictors, performing in terms of different window sizes, time feature encoding schemes, and loss functions. We split data using Leave One Last Item strategy according to [5], aligning with TiSASRec for fair comparison, and use the final

transaction pair ⟨user,item,timestamp⟩ per user for testing, the second last one for validation, and the remaining transactions for training.

Multi-Items Predictor vs. Next-Item Predictor. We compare Model1, the multi-item predictor, with a next-item predictor by asking each model for a set of recommendations and evaluate that whole set on MovieLens-1m, Amazon Beauty, and Amazon Game datasets, e.g., predict $\text{Item}_{t+1} \rightarrow \text{Item}_{t+2} \rightarrow \text{Item}_{t+3}$ with TiSASRec [2] step by step. Then, predict a sequence of [Item_{t+1}, Item_{t+2}, Item_{t+3}] with Model1 that uses the time encoding option 1 by dense all item prediction and compare each performance. We compare the performance of our model in terms of the training objective selection described in Section 3.5 (next-item prediction, all-item prediction, and dense all-item prediction) to the next-item prediction by TiSASRec.

Multi-day Items Predictor vs. Multi-day Items Predictor. We compare Model2 with Model1 that uses the time encoding option 2 i.e., PinnerFormer model [1]. For instance, ask Model2 for a next Item_{t+1} and compare its prediction (Item_{t+1}, +8 day) to the one (Item_{t+8}) by PinnerFormer model. Note: we cannot compare the PinnerFormer model with a next-item predictor or Model1 that uses the time encoding option 1, as it's a multi-day actions prediction model.

4.1 Evaluation Metrics

We adopt two Top-N metrics from our baseline, Hit Rate@10 and NDCG@10, to evaluate recommendation performance. Hit@10 counts the rates of the ground-truth items among the top 10 items. NDCG@10 considers the position and assigns higher weights to higher positions. Following [17], for each user u, we randomly sample 100 negative items, and rank these items with the ground-truth item. We calculate Hit@10 and NDCG@10 based on the rankings of these 101 items. The results are interpreted as the higher the score is, the better the performance. We intent to analyze the results differently by dataset, e.g., low NDCG@10 on Amazon Beauty dataset does not necessarily mean failure of our model, as the average sequence length is short.

5 Expected Outcomes

We aim to demonstrate the superiority of our multi-day window predictor over the traditional next-item predictor. Our proposal to extend TiSASRec [2] by modeling a multi-day window version would be an original contribution to knowledge. We combine the best time-interval encoding scheme and loss function to design a timing-aware multi-day window self-attention mechanism that predicts the next x-items with the best timing. Our work fills gaps in existing research and contributes to debates on the effectiveness of predicting a customer's future purchases compared to predicting their next purchase. We plan to conduct thorough experiments to study the impact of multi-day window prediction, time interval encoding schemes, and different loss functions on our model's performance and extend our understanding of sequential recommendation.

6 Research Plan, Milestones and Deliverables

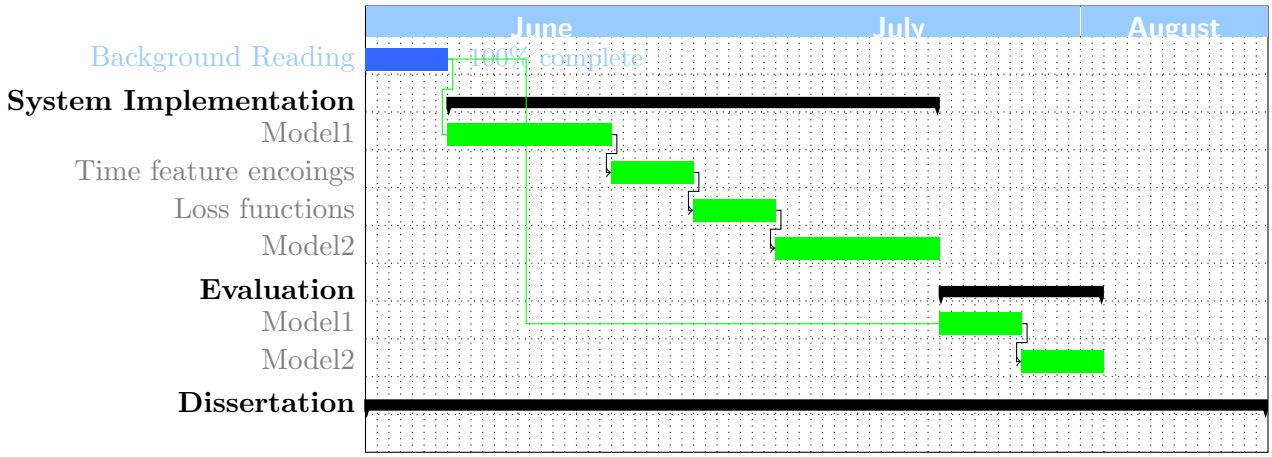

Figure 1: Gantt Chart of the activities defined for this project.

Milestone	Week	Description
M_1	6	System Implementation completed
M_2	8	Evaluation completed
M_3	10	Dissertation submitted

Table 2: Milestones defined in this project.

Deliverable	Week	Description
D_1	4	Model1: Time-interval aware multi-item window predictor
D_2	6	Model2: Timing aware multi-day window predictor
D_3	8	Evaluation report on training objectives, time encodings, and loss functions
D_4	10	Dissertation

Table 3: List of deliverables defined in this project.

References

[1] Nikil Pancha, Andrew Zhai, Jure Leskovec, and Charles Rosenberg. Pinnerformer: Sequence modeling for user representation at pinterest, 2022.

[2] Jiacheng Li, Yujie Wang, and Julian McAuley. Time interval aware self-attention for sequential recommendation. WSDM '20, page 322–330, New York, NY, USA, 2020. Association for Computing Machinery.

[3] Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation, 2018.

[4] Seyed Mehran Kazemi, Rishab Goel, Sepehr Eghbali, Janahan Ramanan, Jaspreet Sahota, Sanjay Thakur, Stella Wu, Cathal Smyth, Pascal Poupart, and Marcus Brubaker. Time2vec: Learning a vector representation of time, 2019.

[5] Zaiqiao Meng, Richard McCreadie, Craig Macdonald, and Iadh Ounis. Exploring data splitting strategies for the evaluation of recommendation models, 2020.

[6] Hui Fang, Danning Zhang, Yiheng Shu, and Guibing Guo. Deep learning for sequential recommendation: Algorithms, influential factors, and evaluations, 2020.

[7] Ivan Bilan and Benjamin Roth. Position-aware self-attention with relative positional encodings for slot filling, 2018.

[8] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2016.

[9] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention, 2016.

[10] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, and Jun Ma. Neural attentive session-based recommendation, 2017.

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[12] Shuai Zhang, Yi Tay, Lina Yao, and Aixin Sun. Next item recommendation with self-attention, 2018.

[13] Fuyu Lv, Taiwei Jin, Changlong Yu, Fei Sun, Quan Lin, Keping Yang, and Wilfred Ng. Sdm: Sequential deep matching model for online large-scale recommender system, 2019.

[14] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks, 2015.

[15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[16] Xinyang Yi, Ji Yang, Lichan Hong, Derek Zhiyuan Cheng, Lukasz Heldt, Aditee Kumthekar, Zhe Zhao, Li Wei, and Ed Chi. Sampling-bias-corrected neural modeling for large corpus item recommendations. In *Proceedings of the 13th ACM Conference on Recommender Systems*, RecSys '19, page 269–277, New York, NY, USA, 2019. Association for Computing Machinery.

[17] Yehuda Koren. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, page 426–434, New York, NY, USA, 2008. Association for Computing Machinery.

A Appendix

A.1 Model Framework

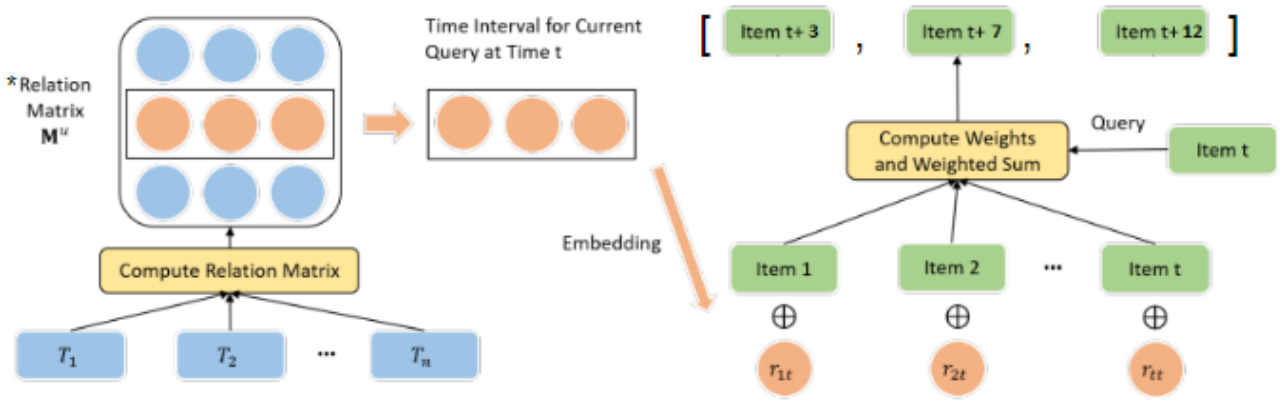

Figure 2: The overall model framework adopted from TiSASRec [2], demonstrating the components corresponding to time intervals and self-attention and an example output. Instead of a next item $t+1$, the model outputs a sequence of items that the user will purchase based on a given sequence of purchased items. *Relation matrix $\boldsymbol{M}^u$ will be replaced with a matrix with action vectors $\boldsymbol{A}^u$ when experimenting time encoding schemes.

A.2 Training Objectives

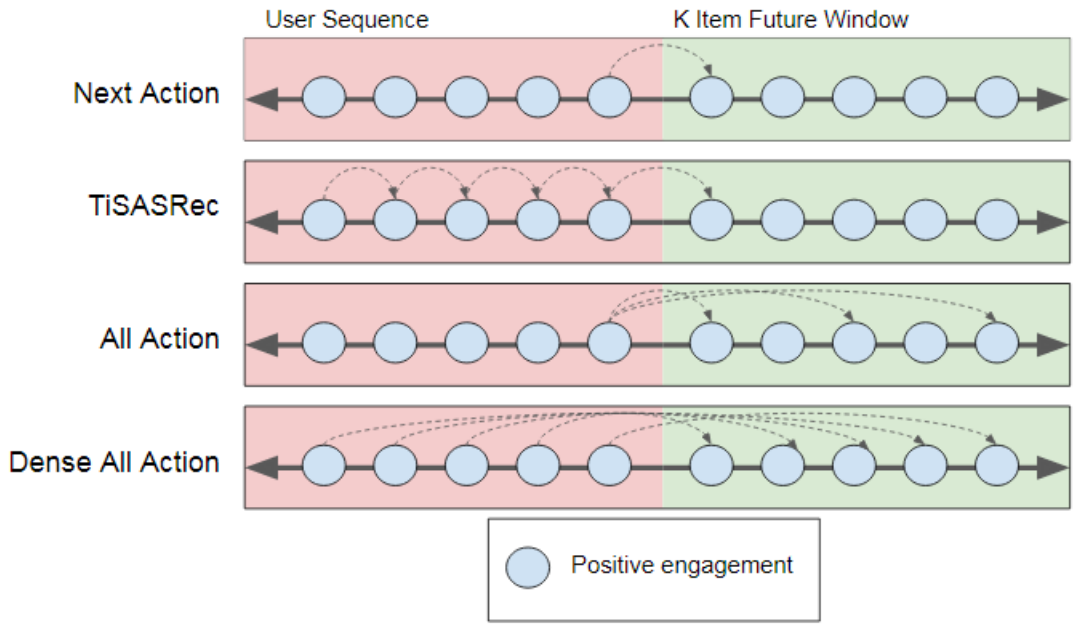

Figure 3: Four training objectives adopted from PinnerFormer [1]. Blue circles represent embeddings corresponding to items as positive engagement. Since our input sequences are purchased items, we don't have negative engagements. The exact pairings in the dense all action loss are sampled, so this is simply one potential materialization.