School of Informatics

Informatics Project Proposal
Time-Interval Aware Multi-day Window Recommender

B231968
April 2023

Abstract

A sequential recommender system widely employs a next-item prediction approach that predicts the next item based on a user's behavior trajectories. Recently the model that considers time intervals between items has shown promising performance. A new approach has also emerged where the model predicts all the items the customer will purchase within a given time window, e.g., the next 5 mins, 1 day, 1 week, etc., - in theory - better predicting how the user preferences will change over time. We suggest evaluating the theory on diverse public datasets extending the existing model while exploring time encoding schemes. We propose a Time Interval-Aware Multi-day Window Self-Attention Recommender that extends the next-item predictor into a multi-day window predictor while modeling time intervals between every item.

Date: Tuesday 18th April, 2023

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

1 Motivation

A sequential recommender system is a next-action prediction system that takes a user's behavior trajectories as input and then adopts recommendation algorithms to recommend appropriate items or services to the user. Traditionally, Markov Chains (MCs), and recently, Recurrent Neural Networks (RNNs) and Self-Attention (SA) have grown due to their ability to capture the dynamics of sequential patterns. Recently next-item prediction using timestamps accounted for time intervals between items has shown promising performance [1]. Due to the successful adaptation of recommendation systems to Transformers, a new theory was also discussed that future-item prediction is more performant than next-item prediction [2]. However, the former time-interval aware model is for next-item prediction, and the latter theory is unclear in its generalized performance on diverse datasets. Also, their effective encoding schemes for time feature is unexplored. In this proposal, we suggest extending the next-item predictor into a multi-day window predictor while modeling the timestamps of interactions within a sequential modeling framework using two different encoding schemes to explore the influence of time intervals on future-item prediction. We propose a Time Interval-Aware Multi-day Window Self-Attention Recommender, which models the absolute positions of items and the time intervals between them in a sequence and captures the user's long-term interest.

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. It might not be best to use a next-item predictor to recommend a face cream consumed once a month in this case. The low performance of SASRec [3] and TiSASRec [1] on the Amazon Beauty dataset might reflect the case. We need to fill the gap between the results predicted by the two strategies. The previous sequential recommendation techniques use the former strategy because they don't consider the user's interest in the long-term timeframe, and we need to solve this problem.

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 time-interval aware multi-day window self-attention mechanism to address these limitations. The model predicts a sequence of multi-items 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-item 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 [1] 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 the proposed research at the most suitable timeline. We extend TiSASRec (2020) [1], one of the latest models for sequence recommendation that considers the time interval between items and exhibits promising performance, and implement it based on the theory of a sequence of items prediction discussed in PinnerFormer (2022) [2]. We have studied the related work and identified that TiSASRec is limited to a next-item prediction, and no work evaluated the theory by Pinnerformer for general use. However, no research has extended the existing model to use on public datasets and investigated the theory and timestamp encoding scheme. Furthermore, our stretch goal in Methodology has the potential to sophisticate the model where it predicts "when and what" to recommend as a human sales representative does. No model has achieved this objective. Therefore, our research is of great novelty as we create a new paradigm in personalized sequential recommendation.

1.4 Significance

We propose a few structural changes on a next-item predictor architecture to implement a multi-day window predictor via a latent factor model. In the next item prediction model [3, 1], a predicted item Item_{t+1} does not include day-factor or time-factor, i.e., predict Item_{t+b} means $t+b$th item, not the item on day $t+b$. This makes it non-trivial to extend the window length simply for this model if we want to predict the item on day $t+b$. Since the Transformer achieves state-of-the-art results and has become a default choice for many sequential recommendation problems, it is interesting to adapt and explore the idea of timing-aware multi-day window prediction for it, and, to our knowledge, no way of doing this has been proposed so far. Common application domains of future items recommendation include e-commerce, POI (Point-of-Interest), music (e.g., Last.fm3), and movie/video (e.g., MovieLens4). Building on our findings, one possible direction for future work might be experimenting with more time modeling. We notice that when a user does not have enough length of purchase history, for example, with the Amazon Beauty dataset, the performance of the model degrades a lot [1]. Also, a longer window is not applicable for such a dataset. However, in real life, for example, a good seller can recommend the best seasonal personalized eye-shadow based on a purchase history with 3-4 make-up items, as one can narrow down customer's purchase traits by knowing a few items out of a hundred choices and when those are purchased. Thus, we think it's worth exploring time modeling, where the model predicts the timing to recommend even with a little purchase history.

1.5 Feasibility

The proposed pipeline of a time-interval aware multi-day window item recommendation primarily consists of timestamp encoding and multi-item training processes. We have identified several alternative machine learning models in the related work and methodology sections. Code implementations of Time2vec [4] and TiSASRec [1] are publicly available. We can use them to create our main pipeline without building and training the model from scratch. We can explore many heuristics relevant to the proposed methodology in training and tuning processes and thus produce several models for our system. We will explore them depending on our progress to ensure the workload is appropriate and we can complete work within the 10-week time frame. All of these suggest that we can complete the project in due time.

1.6 Beneficiaries

The proposed work will benefit the users of the system and the research field of sequential recommendation and personalized machine learning. The system will serve as a tool to help a wide range of users in academia and industry, such as researchers, developers, or e-commerce marketers, who are interested in recommending or predicting future trends to predict items or actions that users will be interested in/engaging while with more control. The recommendations made by the system might provide the user with commercial profits. Besides, we identify the limitations of existing models in the literature and propose two ways to predict future items as our primary goal and the stretch goal of the project. It provides other researchers with a way to track the current progress in the literature and allows them to continue work in this scope. Our proposed model could inspire further innovations in future items recommendation.

2 Background and Related Work

2.1 Sequential Recommendation

A sequential recommendation system is a machine learning system that uses the user's behavior sequence to predict the next item the user would be interested in by providing a ranked item list. Conventional popular methods for the sequential recommendation include frequent pattern mining, K-nearest neighbors, Markov chains (MCs), matrix factorization, and reinforcement learning while Deep Learning methods include Recurrent neural networks (RNNs), Convolutional neural networks (CNNs), Multi-layer perceptrons (MLPs), Attention mechanisms, Self Attention (SA), and Graph neural networks (GNNs) [5]. Traditionally, MCs, and more recently RNNs and SA, have grown due to their ability to capture the dynamics of sequential patterns. 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. On the other hand, RNNs perform better in learning users' long-term semantics with denser datasets, where higher model complexity is affordable. However, these methods have limitations where one's strengths become another's weaknesses. Inspired by the successful application of the SA model to the next item predictor [3], the multi-day window SA predictor has also been applied successfully to predict items after x-days, capturing the user's long-term interest [2]. In this proposal, we present a time-interval aware, multi-day window recommendation system using self-attention mechanism that can learn the users' long term interest using the time gap between every items in their action history. Our work is based on the next item prediction algorithm [3] and similar ideas can be found in [1, 2].

2.2 Attention Mechanisms

We can intuitively understand attention mechanisms in deep learning 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. [6] 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 [7] and image captioning [8]. In sequential recommendations, vanilla attention is proposed and widely used by applying for this work as a decoder for RNNs [9].

2.3 Self-Attention Mechanisms

Self-attention mechanisms are also of growing interest in sequential recommendations. Google initially developed the self-attention mechanism as Transformer in 2017 [10], a pure attention-based sequence-to-sequence technique, which achieved state-of-the-art performance on neural machine translation tasks. Self-attention was then introduced in the sequential recommendation, estimating the weight of each item in the user's interaction trajectory to learn a more accurate representation of their short-term intentions, using a metric learning framework to learn long-term interest. The self-attention mechanism does not involve RNN structures in contrast to the attention mechanism but performs much better than RNN-based models [11]. In SDM [12], a multi-head self-attention module is incorporated to capture a user's multiple interests in a session (i.e., short-term preference), while the long-term interest of the user is also encoded through attention and dense-fully-connected networks based on various types of side information, e.g., item ID, first level category, leaf category, brand and shop in historical transactions of users. SASRec [3] employs a self-attention layer to predict the next item from sequences of purchase history of users using only item IDs from the product/movie review datasets. Since the self-attention model doesn't include any recurrent or convolutional module, it is not aware of the positions of previous items. One solution is to add positional encodings to the inputs, which can be a deterministic function or learnable positional embedding [13, 10]. Another solution uses relative position representations [6]. They model the relative positions between two input elements as pairwise relationships. Inspired by self-attention with relative positions, TiSASRec [1] combines the absolute positions and relative positions to design time-aware self-attention, which models items' positions and time intervals, improving SASRec. Recently, a new theory emerged due to performance improvement in the next-item prediction by Pinnerformer [2]. They extends SASRec and attempts to predict actions multiple days ahead rather than predict the next item, capturing the long-term interest of users. Inspired by PinnerFormer, our model uses TiSASRec as a baseline and predicts multiple items ahead while considering the time interval between every item, aiming to explore the theory further and improve model performance.

3 Programme and Methodology

3.1 Time Interval-Aware Multi-day Window Self-Attention Recommender

As a main goal of the project, we propose a Time Interval-Aware Multi-day 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 our model is to 1. provide a ranked item list and establish its performance as a multi-day 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 [1]: (a) extend the baseline into a multi-item window predictor, (b) experiment with two encoding schemes of time feature and replace it with the best one, and (c) experiment with binary cross-entropy and sampled softmax [2] and replace with the best loss function.

3.2 (a) Multi-day Window Predictor

The problem formulation is 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. Our model's input is $(S_1^u, S_2^u, ..., S_{|S^u|-1}^u)$ and time intervals R^u between any two items 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. To predict the Item_{t+b} where b is the target item, we first get the combined representation of items, positions, and time intervals from stacked self-attention blocks. We use it to compute users' preference score of item i, using a latent factor model: $A_{i,t} = \boldsymbol{Z}_t \boldsymbol{M}_i^I$, 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.

3.3 (c) Loss Function

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 [14], 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 instance, $o_1 = s_{t+7}$ if $1 \leq t+7 < n$, describing the output for the item at $t+7$ in the window. Since the loss is computed using negative samples and the sampling approach affects performance, we might also explore some sampling methods.

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 [1], $\|\cdot\|_F$ 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 [2], we try adopting their sampled softmax with a $\log Q$ correction [15] 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.

3.4 Training Objectives.

We adopt three training objectives from [1, 2] that are described below, and are depicted in Figure 3 in Appendix, aiming to achieve using the user's action sequence $(S_1^u, S_2^u, ..., S_{|S^u|-1}^u)$.

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., $(\text{Item}_{t+3}, \text{Item}_{t+8}$, and $\text{Item}_{t+12})$, 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. Then, at inference time, we use e_t as our final user representation.

3.5 (b) 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.

Time Interval as a Square Relation Matrix.

To represent the personalized time interval, we adopt [1] 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.

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

Time Interval as a Matrix of Action Vectors. To represent the time an action occurred, we adapt [2] 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 [2] 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.

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

3.6 Timing-Aware Multi-day Window Self-Attention: Stretch Goal

The limitation of the main goal is that it only models what to recommend in whici order and cannot recommend it in a timely manner as a human sales representative does. As a stretch goal 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.

3.7 Risk Assessment

We assess the possible risks of the project in Table 1 and describe them in order. 1. Implementation trouble is the highest risk, as it prevents us from making progress on the project from the starting point. However, since the primary modification we make on the existing model is to change the window size, i.e., the position of input embedding, we can minimize the risk by making sure the reproduction of the baseline before the project starts. This would not be difficult as the baseline code is available online [1]. 2. It’s likely that our extended model does not work at the first point because the length of the user’s action sequence could be too short/too long while we play with the window size of the sequence. To establish the model performance, we can try different datasets or experiment with window size depending on the dataset. 3. Encoding time interval as a matrix of action vectors, described in Section 3.4 could be struggling, as PinnerFormer [2] does not publish code. Fortunately, Time2vec [4] publish code and we can extensively review the paper and its implementation in the background reading phase to reproduce the encoding scheme. 4. Finally, the sampling softmax loss function we try to adopt from [2] may not work, as we use different architecture and datasets. In such a case, we may use the binary cross-entropy loss function the baseline uses or search for an alternative one, but the baseline would also be enough, as it already outperforms the next item prediction. Thus, as for 3 and 4, it would probably make sense to pick the baseline scheme and use the time for the stretch goal rather than 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	Our model not working	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

There are no ethical concerns in our experiments with public datasets. Also, since the users of our model would be researchers and corporate developers, the code would not directly harm them. However, depending on the types of datasets and the use case, the information they provide to end-users (consumers) may be biased as the result of recommendations. For example, the model may expose only information about shoes if the purchase history of a woman on an e-commerce website is shoes only, and she may not explore the possibility of her interest in cosmetic products. Firms integrating our model into their system should consider such biases before using them. Besides, we will follow the School of Informatics ethics procedure and submit an Informatics Ethics Form for review to ensure that our study complies with the General Data Protection Regulation and the policy of the University of Edinburgh.

4 Evaluation

We extend our baseline [1], a next-item predictor, into a multi-day window predictor and compare it with the baseline on MovieLens-1m, Amazon Beauty, and Amazon Game datasets. Specifically, we compare the performance of our model in terms of the training objective selection described in Section 3.4 (next-item prediction, all-item prediction, and dense all-item prediction) to the next-item prediction by TiSASRec. For instance, follow these steps: (1) Predict Item_{t+3} with TiSASRec by predicting $\text{Item}_{t+1} \rightarrow \text{Item}_{t+2} \rightarrow \text{Item}_{t+3}$ step by step. (2) Predict Item_{t+3} with our model by all item prediction and dense all item prediction. (3) Evaluate each performance. We perform this evaluation in terms of different window sizes, time feature encoding schemes, and loss functions.

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 [16], 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 hope to prove that our multi-day window predictor outperforms the traditional next-item predictor. Our proposal to extend the window size of a user's action sequence and model a multi-day window version of TiSASRec [1] would be a novel idea, as PinnerFormer [2] extends the older model SASRec [3], and it should make an original contribution to knowledge. In this work, we combine the best time-interval encoding scheme and the best loss function for self-attention and design a time interval-aware multi-day window self-attention mechanism to learn the weight of different items, absolute positions, and time intervals to predict the next x-items. Those unexplored attempts should fill gaps in existing work and contribute to debates about whether predicting the sequence of a customer's future purchases is a better learning task for recommender systems than the standard approach of predicting their single next purchase. Finally, a thorough experiment to study the impact of multi-day window prediction, time interval encoding schemes, and different loss functions, on the performance of our model should help us extend our understanding of topics in the sequential recommendation.

6 Research Plan, Milestones and Deliverables

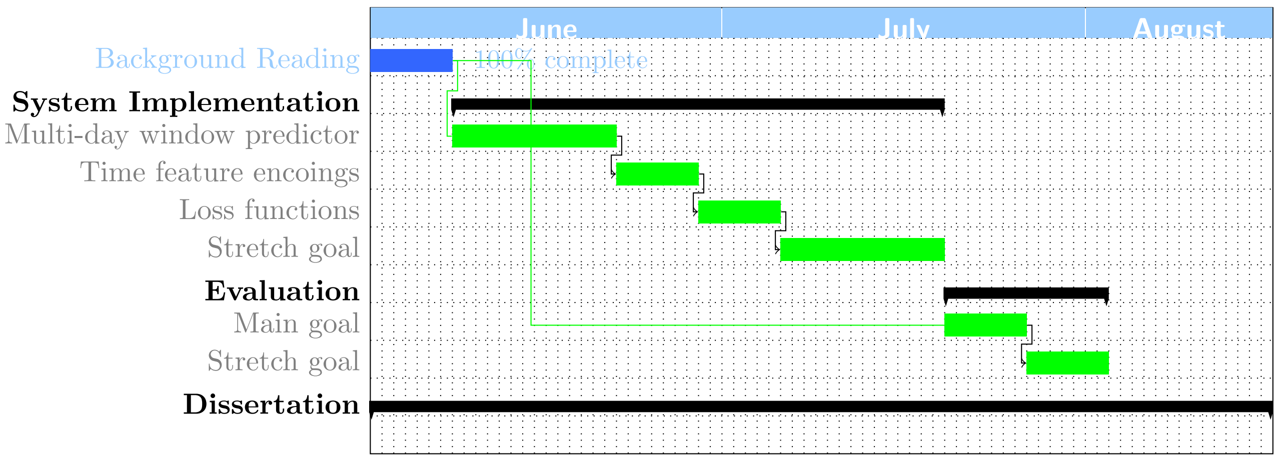

Figure 1: Gantt Chart of the activities defined for this project.

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	Time-interval aware multi-day window predictor
D_2	6	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] 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.

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

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

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

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

[8] 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.

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

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

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

[12] 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.

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

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

[15] 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.

[16] 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

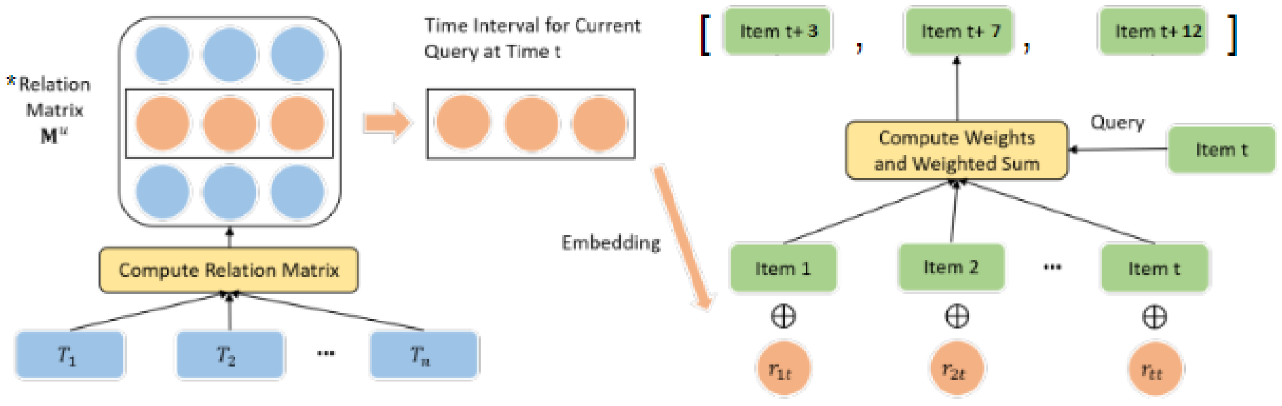

Figure 2: The overall model framework adopted from TiSASRec [1], demonstrating the components corresponding to time intervals and self-attention and an example output. Instead of a next item $t+1$, the model outputs a sequence of items that the user will purchase based on a given sequence of purchased 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

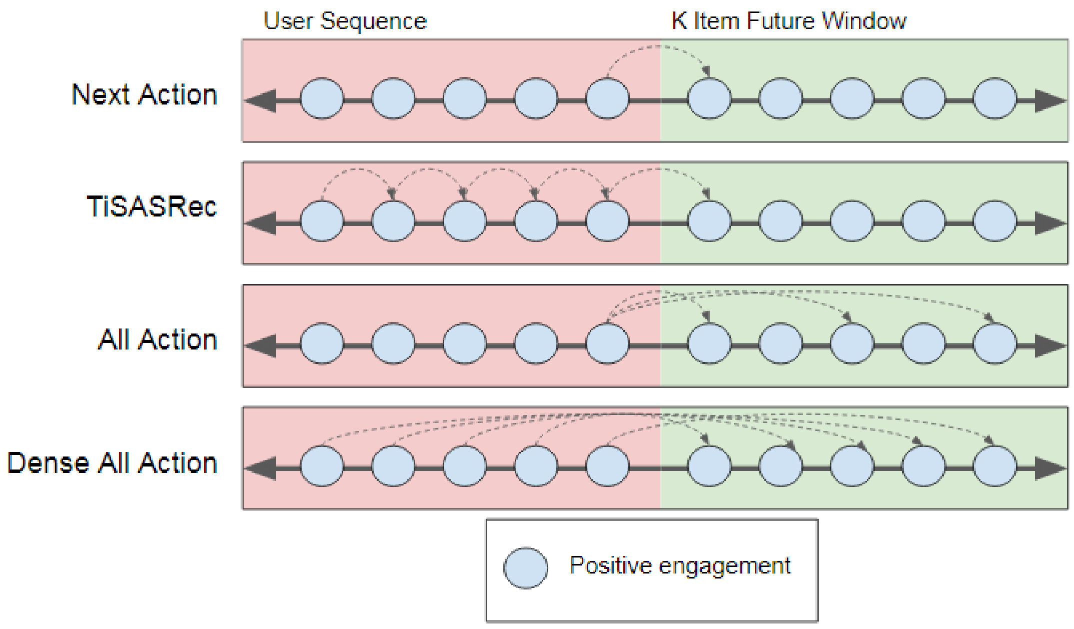

Figure 3: Four training objectives adopted from PinnerFormer [2]. Blue circles represent embeddings corresponding to items as positive engagement. Since our input sequences are purchased items, we don't have negative engagements. The exact pairings in the dense all action loss are sampled, so this is simply one potential materialization.