Incorporating Time in Sequential Recommendation Models

Mostafa Rahmani
rahmani.sut@gmail.com
Amazon
WA, USA

James Caverlee
jacaverl@amazon.com
Amazon
TX, USA

Fei Wang
feiww@amazon.com
Amazon
CA, USA

ABSTRACT

Sequential models are designed to learn sequential patterns in data based on the chronological order of user interactions. However, they often ignore the timestamps of these interactions. Incorporating time is crucial because many sequential patterns are time-dependent, and the model cannot make time-aware recommendations without considering time. This article demonstrates that providing a rich representation of time can significantly improve the performance of sequential models. The existing literature treats time as a one-dimensional time-series obtained by quantizing time. In this study, we propose treating time as a multi-dimensional time-series and explore representation learning methods, including a kernel based method and an embedding-based algorithm. Experiments on multiple datasets show that the inclusion of time significantly enhances the model's performance, and multi-dimensional methods outperform the one-dimensional method by a substantial margin.

KEYWORDS

Sequential models, Time-aware recommendation, Time-series

ACM Reference Format:
Mostafa Rahmani, James Caverlee, and Fei Wang. 2023. Incorporating Time in Sequential Recommendation Models. In *Seventeenth ACM Conference on Recommender Systems (RecSys '23), September 18–22, 2023, Singapore, Singapore.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3604915.3608830

1 INTRODUCTION

Sequential recommendation systems are the backbone of many popular platforms, helping users connect to the right content. By modeling the order of user-item interactions, these systems typically optimize for the next interaction in a sequence (be it movie to stream, song to listen to, and so on) [7, 8, 10, 15, 20, 21, 25]. A fundamental research challenge in these and related methods is: how to best model the temporal sequence of user-item interactions? Most existing sequential approaches make one of two assumptions:

- The user-item interactions are viewed solely as chronological sequences without fine-grained interaction times. That is, the sequence ($item_A \rightarrow item_B \rightarrow item_C$) considers only that the three items were consumed in order not *when* the consumption happened. This assumption is the foundation of many popular methods, e.g., [8, 10, 21].
- Alternatively, a few recent works explicitly model time as a unidimensional variable in sequential recommendation, e.g., [3, 13, 24]. In this unidimensional assumption, a reference time t_0 is set equal to an old time or the first time-stamp in the data. Next, each time-stamp t is represented by the time delta between t and t_0 quantized in seconds (or hours or days, depending on the specific scenario).

Although sequential recommendation systems have shown strong performance, these two assumptions may lead to undesirable outcomes. For the first assumption, a sequential model that ignores time may miss important contextual patterns that are intrinsically time-dependent. For example, a user who streams an action movie on a Saturday night may have quite different preferences later that same night versus Sunday morning. Similarly, a user who purchases an air purifier may be expected to purchase a new filter after six months, regardless of the specific order of item interactions. For the second assumption, a unidimensional representation of time may not capture distinct user behaviors at different times (e.g, viewing short news videos over breakfast in the morning versus longer-form entertainment in the evening), periodic consumption behaviors (e.g., on a daily or weekly cycle), and other important contexts.

Further, the careful modeling of time in sequential models has the added benefit of helping to explain many interactions. To illustrate, consider a movie streaming service that adds a new popular movie to the system at a specific time. If the time of interaction is not provided to the model, the model will be unable to explain why the movie has been watched with many irrelevant titles. Without this temporal information, the model cannot learn from interactions that are influenced by the movie's popularity at that specific time. As a result, these interactions may be considered as noisy labels and could even hinder the model's learning process if used to define training tasks.

Hence, in this paper, we explore new approaches for temporal modeling in sequential recommendation. Our investigation considers four time-related pieces of information: (i) the current time-stamp (query time); (ii) the time-stamp of the previous interactions; (iii) the time-difference between previous interactions; and (iv) the time-difference between the current time-stamp and the previous interactions. Concretely, we propose a *multidimensional-based representation learning* method designed to carefully leverage these four time-related pieces of information. We consider both a simple linear embedding approach as well as a data-driven kernel. Through experiments on two well-known benchmark datasets and two proprietary recommendation datasets, we find that (1) integrating such rich models of time significantly boosts model accuracy; and (2) the

RecSys '23, September 18–22, 2023, Singapore, Singapore
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0241-9/23/09…$15.00
https://doi.org/10.1145/3604915.3608830

proposed multidimensional-based representation learning methods notably outperform those that treat time as a unidimensional time-series.

2 RELATED WORK

The goal of a sequential recommendation model is to predict the next interaction in a sequence. Two of the early solutions based on deep neural networks are GRU4rec [8] and GRU4Rec+ [7] which use a Recurrent Neural Network architecture to build the sequence encoder. Recent state-of-the-art models use attention and transformers [15] to build the sequence encoder. SASRec [10] leverages transformer blocks to predict the next item in the sequence based on all previous elements. BERT4Rec [21] employs the bidirectional training technique used in the well-known BERT language model [2]. In bidirectional training, each time when a user sequence is used for training, a random subset of the items are replaced with a mask token (or random tokens) and the model is asked to retrieve the masked items. Since BERT4Rec is a state-of-the-art sequential model, [14], we use it as the base sequential model in our experiments. Recently, several studies have employed graph neural networks and knowledge graphs as demonstrated in works such as [3, 4, 9, 16–18, 22]. These models typically leverage additional information such as cross-session connections, item knowledge graphs, or graphs that connect related users.

Most sequential recommendation models in the literature only consider the chronological order of interactions and make recommendations that are independent of the query time. However, a few studies have explored leveraging the interaction time. For example, [13, 24] suggest making the attention matrices computed by the transformer module a function of the time-difference between interactions. [3] proposes a technique similar to that used in [23] to compute a continuous representation of time (the employed technique is equivalent to Time2vec in [11]). We refer to these approaches as unidimensional-based methods because they treat time as a unidimensional time-series. In our experiments, we compare against the approaches used in [3, 11, 23] to demonstrate the significance of decomposing time-stamps into a multivariate time-series.

3 MULTI-DIMENSIONAL TEMPORAL METHODS

Our core modeling assumption is that important temporal patterns can be attributed to factors such as hour-of-day, day-of-week, week-of-month, and month-of-year. For example, horror movies may be typically viewed in the late hours, and demand for electric heaters is highest in December. To more accurately capture such temporal patterns, we break down each time-stamp into multiple meaningful components: e.g., year, month-of-year, week-of-month, day-of-week, hour-of-day, and so on.

3.1 Formal Model and Challenges

Concretely, we denote the multi-dimensional time-stamp vector as $\mathbf{t} \in \mathbb{R}^d$, the unidimensional representation of time as t, and the exact time-stamp as τ. Each vector $\mathbf{t}$ is created by decomposing its corresponding time-stamp τ. We also normalize each element of $\mathbf{t}$ as $\mathbf{t}(i) = \frac{\mathbf{t}(i) - \zeta_{\min} + 1}{\zeta_{max}}$, where $\zeta_{\min}$ and $\zeta_{\max}$ are the minimum and maximum values that $\mathbf{t}(i)$ can have. For example, if $\mathbf{t}(i)$ represents the day-of-week, $\zeta_{\min} = 1$ and $\zeta_{\max} = 7$.

This multidimensional representation of time is then used as the input for a time-stamp encoder module, which maps the multidimensional time-stamp vector to a higher dimensional embedding space. A key challenge is how to design such an embedding mapping. We present two mapping functions to tackle this challenge: 1) the first is based on kernel approximation theory using random projections, which has the benefit of enabling the model to have an estimate of time-differences between different time-stamps; and 2) the second uses linear embedding to map each component of the time-stamp vector, which lets the model learn optimal representation for different components of the time-stamps.

In Figure 1, we showcase the effective utilization of computed representation vectors from the time-stamp encoder in a transformer-based sequential model during inference. The model takes two inputs: a sequence of past interactions augmented with a mask token, and a sequence of time-stamps for these interactions appended by the query time. During training, we provide both the sequence of past interactions and their corresponding time-stamps (the time-stamps of masked tokens serve as query time for those tokens). The time-stamp encoder's output is combined with the output from the item embedding layer, creating an aggregated input that is then fed into the sequence encoder.

In this study, we adopted BERT4Rec [21] as our baseline sequential model due to the mask token's presence, which facilitates incorporating query time in the model's input. In Section 4, we delve into several techniques that empower alternative sequential models to integrate both the query time and the timing of past interactions.

Contribution: The proposed method of time-stamp decomposition, while simple, has not been studied extensively in previous works. To the best of our knowledge, this is the first study to propose multiple representation learning approaches (e.g., a kernel approximation based method and an embedding-based approach) based on decomposed time-stamps. We explicitly compare unidimensional vs multidimensional temporal approaches and experimentally confirm the advantages of using rich multi-dimensional methods. In addition, we introduce techniques for effectively utilizing query time in sequential models.

3.2 Projection-based approach

The first approach to map the multidimensional time-stamp vector to a higher dimensional embedding space is based on random projections. Time provides two important pieces of information to the model: when the interactions happened, and what were the time-differences between different interactions. Therefore, not only should one be able to accurately estimate the time-stamp from its representation, but also the model should be able to have an estimate of the time-difference between interactions via comparing their time-stamp representation vectors. Since most deep learning based methods use inner-product as the measure of similarity, we desire a representation function $\phi(\mathbf{t}) \in \mathbb{R}^{d'}$ such that

$$\phi(\mathbf{t}_1)^T \phi(\mathbf{t}_2) \approx \kappa(\mathbf{t}_1 - \mathbf{t}_2) , \quad (1)$$

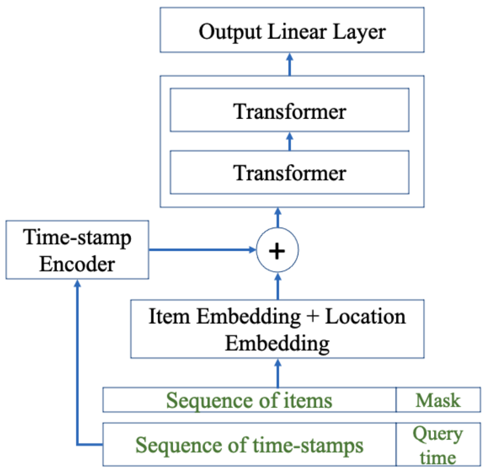

Figure 1: The structure of the model used in the presented experiments. The model is similar to the state-of-the-art model used in [14, 21] with the addition of the time-stamp encoder. The dimension of data at the output of the embedding layer is $\mathbb{R}^{b\times n\times d'}$ where b is the batch size and n is the length of the sequence. Similarly, the dimension of data at the output of the time-stamp encoder is $\mathbb{R}^{b\times n\times d'}$. These two outputs are added and fed into the transformers.

where $\kappa(\cdot)$ is a positive definite shift-invariant kernel. In other words, the mapping function $\phi(\cdot)$ enables the model to infer the distance between each pair of interactions via simply computing the inner-product between their time-representation-vectors. This is specifically a useful feature for transformer based models since their attention modules use the inner-product to compare each pair of items (or tokens) in the sequence. In the following, we review the theory of random projection based kernel approximation which inspires us to design the mapping function $\phi(\cdot)$.

3.2.1 Approximating non-linear kernel functions via random projections. The authors of [19] showed that the value of a non-linear kernel function $\kappa(\mathbf{x},\mathbf{y})$ can be approximated via the inner-product of the vector of the random projection values. The following Algorithm 1 and Theorem 1 provide the details.

Algorithm 1 Random Fourier Features

```
Input: Positive definite shift-invariant kernel $\kappa(\mathbf{x}-\mathbf{y})$, $\mathbf{x}\in\mathbb{R}^d$, and $\mathbf{y}\in\mathbb{R}^d$.
1. Define $\Omega$ as the Fourier transform of the kernel function
    $\Omega(\mathbf{w}) = \frac{1}{2\pi}\int \exp(-j\mathbf{w}^T\delta)\kappa(\delta)d\delta$ .
2. Sample $\{\mathbf{w}_i\in\mathbb{R}^d\}_{i=1}^{d'}$ from $\Omega$ independently and sample $\{b_i\}_{i=1}^{d'}$ from the uniform distribution on $[0,2\pi]$.
3. Define $\phi(\mathbf{t})\in\mathbb{R}^{d'}$ as
    $\phi(\mathbf{t}) = [\cos(\mathbf{w}_1^T\mathbf{t}+b_1), \cos(\mathbf{w}_2^T\mathbf{t}+b_2)\ ,\ \dots,\ \cos(\mathbf{w}_{d'}^T\mathbf{t}+b_{d'})]$ .
```

Theorem 1 (Adapted from [19]). *Let $\mathcal{M}$ be a compact subset of $\mathbb{R}^d$ with diameter diam$(\mathcal{M})$. Then, for the mapping function $\phi(\cdot)$ defined in Algorithm 1, we have*

$$\mathbb{P}\left[\sup_{\mathbf{x},\mathbf{y}\in\mathcal{M}} |\phi(\mathbf{x})^T\phi(\mathbf{y}) - \kappa(\mathbf{x}-\mathbf{y})| > \epsilon\right] < 8\left(\frac{\sigma_\Omega\, diam(\mathcal{M})}{\epsilon}\right)^2 \exp\left(-\frac{d'\epsilon^2}{4(d+2)}\right), \tag{2}$$

where $\sigma_\Omega^2 = \mathbb{E}_\Omega[\mathbf{w}^T\mathbf{w}]$ is the second moment of the Fourier transform of the kernel function κ.

According to Theorem 1, if d' is sufficiently large, Algorithm 1 offers a precise approximation of the kernel. However, selecting the optimal kernel function for a given task and computing its Fourier transform can be challenging or even impossible in practice. In lieu of this, a more practical approach is for the sequential model to learn the best directions $\{\mathbf{w}_i\}_{i=1}^{d'}$ for the task at hand. We address this by treating the projection directions $\{\mathbf{w}_i\}_{i=1}^{d'}$ and the bias values $\{b_i\}_{i=1}^{d'}$ as the sequential model's parameters to be learned during the training process, alongside the other model parameters. In this way, the model can learn the projection directions that are optimal for the given task or dataset.

3.3 Embedding-based approach

The second approach maps the multidimensional time-stamp vector to a higher dimensional embedding space by learning the embeddings during training. Recall that each time-stamp is decomposed into several components where each component is a discrete value. For instance, day-of-week is a discrete variable with values between 1 to 7. Thus, each value of the time-stamp vector $\mathbf{t}\in\mathbb{R}^d$ can be described using a one-hot vector. Let us define $\mathbf{e}_i\in\mathbb{R}^{m_i\times 1}$ as the one-hot vector corresponding to the i^{th} element of $\mathbf{t}$ and m_i is the number of distinctive values that $\mathbf{t}(i)$ could have (if $\mathbf{t}(i)$ represents day-of-week, then $m_i = 7$). In addition, let us define learnable embedding matrices $\{\psi_i\}_{i=1}^d$ such that $\psi_i\in\mathbb{R}^{m_i\times r_i}$. Each embedding dimension r_i is chosen proportional to the value of m_i, i.e., $r_i = d'\frac{m_i}{\sum_i m_i}$. Therefore, the final mapping function is defined as

$$\phi(\mathbf{t}) = \left[\mathbf{e}_1^T\psi_1\ ,\ \mathbf{e}_2^T\psi_2\ ,\ \dots,\ \mathbf{e}_d^T\psi_d\right]^T \in\mathbb{R}^{d'\times 1}. \tag{3}$$

The embedding matrices, denoted as $\{\psi_i\}_{i=1}^d$, are considered as parameters of the sequential model, which are learned along with the other model parameters.

Remark 1. *Theoretically, the embedding based method can converge to embedding matrices such that the final mapping function is equivalent to the mapping function of a projection-based method. Thus, in theory, the embedding based method should perform at least as well as the projection-based method. However, in the projection-based method, we directly impose the kernel approximation structure which can help the learning algorithm to converge to a proper mapping function even if a sufficient amount of training data is not available. As a result, choosing the right method is task specific. We empirically compare the two and find that the projection based method mostly outperforms when the size of data is small.*

4 EXPERIMENTS

In this section, we empirically investigate the performance of a sequential model with different time embedding methods.

4.1 Setup

Data and Metrics. We use two public datasets and two proprietary ones:

- **MovieLens 20M [5]:** ML-20m is a popular benchmark dataset. After filtering out items with fewer than five interactions and users with fewer than two interactions, the resulting dataset contains 18,166 items, 138,475 users, and around 19.7 million interactions.
- **Amazon Beauty [6]**: Beauty is a well-known dataset that was created by crawling product reviews from Amazon. Once we filtered out items with fewer than five interactions and users with fewer than two interactions, the resulting dataset contains 66,201 items, 266,212 users, and approximately 0.9 million interactions.
- **Proprietary Industry Recommendation (PIR) datasets**: We built two datasets using the data of users in two different marketplaces (i.e., two different countries). In both datasets, for each user, the last 100 interacted items was used to build the sequences. We randomly selected 600k users with sequences longer than 1 and used 500k users for training, 50k for validation, and another 50k for testing.

The MovieLens and Beauty datasets use Unix format timestamps, which measure time as the number of seconds elapsed since 00:00:00 UTC on January 1st, 1970. As this format is a global time representation, it is impossible to determine the local time of users. Hence, we assume that users in these two datasets are in the US/Central time zone and calculated the full timestamps accordingly, with a possible error of less than one day. However, for the PIR datasets, we have access to the full local timestamps. In our experiments, we chose to consider all items for evaluation, which reflects the real-world conditions that the model would encounter in a production setting.[1]

The literature employs two primary evaluation methods: Leave-One-Out and data-splitting strategies. In Leave-One-Out, all users are employed for both training and testing. Each user's final interaction is reserved for testing purposes. For validation purposes, we select the second-to-last action of 2,048 randomly selected users, with the remaining interactions utilized for training. On the other hand, data-splitting strategy divides data into train, validation, and test sets across users. During testing, the model is required to predict the user's last interaction. We applied Leave-One-Out for the MovieLens and Beauty datasets, and data-splitting for the larger PIR datasets in our experiments. For each, we measure the NDCG@k and the Recall@k.

Methods. Since transformers are considered the state-of-the-art approach for sequence encoding [14, 21], the model in our experiments is built using transformer modules. Research has shown that transformer-based models perform best when trained in a bidirectional manner [14, 15], so we used the structure and training algorithm from [21] to construct a next item prediction model for most of the presented experiments. In the following, we present an overview of the methods used in the study. In all of the models that incorporate time, the model receives the output of the time-stamp encoders, as illustrated in Figure 1.

- **Without time**: The output of the time-stamp encoder is not added to the output of the embedding layer. This means that neither the interaction times nor the query time are provided to the model during training and testing. Thus, this baseline is equivalent to the state-of-the-art BERT4Rec model studied in [15, 21].
- **Unidimensional**: This baseline treats time as a unidimensional time-series, which requires us to use quantized time-stamps. While the MovieLens and Beauty datasets already provided quantized time-stamps, for the PIR datasets, we calculated the distance of each time-stamp from the minimum time-stamp in the data and quantized it in hours. For all the datasets, we normalized each value by subtracting the minimum time-stamp value and dividing by the maximum time-stamp value. The mapping function was computed as $\phi(t) = [\cos(g_1 t + b_1), \cos(g_2 t + b_2)\ , \ \dots\ ,\ \cos(g_{d'} t + b_{d'})]$, where $\{g_i\}_{i=1}^{d'}$ and $\{b_i\}_{i=1}^{d'}$ were defined as learnable parameters. This method is equivalent to the algorithm studied in [3, 11, 23].
- **Projection-based method:** This method is described by Algorithm 1 where $\{\mathbf{w}_i\}_{i=1}^{d'}$ and $\{b_i\}_{i=1}^{d'}$ are learnable parameters. In this method, $\mathbf{t} \in \mathbb{R}^5$ (year, month-of-year, week-of-month, day-of-week, hour-of-day) and $\phi(\mathbf{t}) \in \mathbb{R}^{d'}$ was constructed as denoted by Step 3 of Algorithm 1.
- **Embedding-based method:** In this method, $\mathbf{t} \in \mathbb{R}^5$ and $\phi(\mathbf{t}) \in \mathbb{R}^{d'}$ was constructed as in (3).

Training Details. The input embedding layer dimension was set to 256 for experiments using the MovieLens dataset and the Beauty dataset, and it was set equal to 512 for experiments using the PIR datasets. Two consecutive transformer modules were used to create the sequence encoder, and the attention modules had 4 heads. The masking probability in the bidirectional training was set to 0.15, and the dropout probability in the transformer modules was 0.1. The optimizer used was Adam [12]. Figure 1 shows the structure of the model.

4.2 Results

Table 1 compares the performance of the tested methods. The results show that the multi-dimensional based methods outperform the unidimensional based method. For the PIR dataset, only the improvements over the baseline, which did not consider time, are reported. With the PIR datasets, the two methods performed similarly, however, the projection-based method outperforms the embedding-based method on the MovieLens and Beauty datasets. Our hypothesis is that the smaller size of the MovieLens and Beauty datasets allows the projection-based method to converge better to the appropriate mapping, thanks to the kernel approximation structure it imposes. The remarkable performance gains observed in the PIR

[1]In previous studies such as [21], the performance of the model was evaluated on a subset of items, where 100 negative items were randomly sampled for each positive item, and the rankings of the 101 items were used for evaluation. This method, which is based on popularity-based sampling of negatives, has been criticized for its inconsistent results with full ranking evaluations, as shown in recent work [1].

Table 1: The improvement in the performance of BERT4Rec with different methods of incorporating time. N@k and R@k denote NDCG and recall for the top-k recommendations. For MovieLens and Beauty datasets, Results with the best method is also presented.

	N@1	N@10	N@20	R@1	R@10	R@20
Results, without time (ML-20M)	0.076	0.163	0.185	0.076	0.274	0.365
Results, projection based method (ML-20M)	0.086	0.18	0.203	0.086	0.301	0.394
Improv., unidimensional method (ML-20M)	0.99%	3.14%	0.44%	0.99%	0.98%	0.22%
Improv., embedding based method (ML-20M)	2.95%	4.83%	4.95%	2.95%	5.71%	5.77%
Improv., projection based method (ML-20M)	12.21%	10.33%	9.17%	12.21%	9.75%	8.06%
Results, without time (Beauty)	0.0152	0.031	0.035	0.0152	0.051	0.066
Results, projection based method (Beauty)	0.0175	0.033	0.037	0.0175	0.053	0.069
Improv., unidimensional method (Beauty)	3.98%	2.12%	1.35%	3.98%	2.38%	0.27%
Improv., embedding based method (Beauty)	8.28%	0.23%	0.74%	8.28%	-3.35 %	-1.4 %
Improv., projection based method (Beauty)	15.13%	6.45%	6.28%	15.13%	3.92%	4.50%
Improv., unidimensional method (PIR-1)	13.40%	8.00%	6.48%	13.40%	4.65%	3.11%
Improv., embedding based method (PIR-1)	34.40%	20.56%	17.44%	34.40%	11.03%	8.12%
Improv., projection based method (PIR-1)	34.02%	20%	17.28%	34.02%	10.66%	7.78%
Improv., unidimensional method (PIR-2)	7.63%	6.12%	5.91%	7.63%	4.48%	3.18%
Improv., embedding based method (PIR-2)	17.31%	16.61%	16.12%	17.31%	16.03%	14.99%
Improv., projection based method (PIR-2)	14.73%	15.60%	15.25%	14.73%	15.92%	15.01%

datasets strongly suggest that the inclusion of time significantly enhances the model's understanding of the data. Specifically, while the model already processes the sequences to identify item-item relationships, the incorporation of time enables it to also identify item-time relationships and track the popularity of items over time.

Incorporating query time in other sequential models: The method presented can be employed with any sequential recommendation model by adding the output of the time-stamp encoder to the item embedding layer's output. However, unlike BERT4Rec, most other sequential models do not have a mask token that can be utilized to include the query time's embedding vector. As discussed in the next section, it is crucial to include the query time, and either the training algorithm or the architecture needs to be modified to facilitate this. As an example, we explain two ideas of how to modify SASRec to enable it to consume the query time. In the architecture shown in Figure 1, the i^{th} input of the first transformer block is $\phi(\mathbf{t}_i) + \mathbf{e}(x_i) + \mathbf{c}_i$, where $\phi(\mathbf{t}_i)$ is the embedding of i^{th} time-stamp, $\mathbf{e}(x_i)$ is the embedding vector of i^{th} item, and $\mathbf{c}_i$ is the i^{th} learnable location embedding vector [21]. One idea to enable the SASRec model to use query time is to employ two time-encoder modules and define the i^{th} input of the first transformer block as $\phi_1(\mathbf{t}_i) + \phi_2(\mathbf{t}_{i+1}) + \mathbf{e}(x_i) + \mathbf{c}_i$ where $\phi_1(\mathbf{t}_i)$ is the embedding of the i^{th} time-stamp computed by the first time-encoder module and $\phi_2(\mathbf{t}_{i+1})$ is the embedding of the next time-stamp computed by the second time-encoder module. Using this method, the embedding of the query time computed by $\phi_2(\cdot)$ is added to the embedding vector of the last item. Another approach to enable SASRec to incorporate query time is to slightly change the function which prepares the training examples. Specifically, for a given sequence, we randomly select an element from the sequence, remove all subsequent items, and replace the chosen token with the mask token. This facilitates auto-regressive training, similar to the SASRec model and adds a mask token to the sequence. Table 2 demonstrate the enhancements in SASRec model performance through various time embedding methods. These results distinctly indicate that the SASRec model gains a better understanding of the data when time is included in the input. While this study does not compare the performance of different sequential models, it is noteworthy that in experiments conducted on all datasets, BERT4Rec consistently outperformed SASRec.

4.3 Ablation Study: Query time versus the time of former interactions

We observe that incorporating time significantly improves the model's performance. To achieve this, we utilize the time-stamp encoder to generate time embedding vectors for both the query time and the time of former interactions. We conduct an analysis to examine the individual impact of each component, namely, the time of former interactions and the query time. Table 3 summarizes the improvement in the performance of the Bert4Rec model when using the projection-based method on the PIR-1 dataset under three different scenarios: (a) with both query time and time of former interaction, (b) with only the time of former interactions, and (c) with only query time. The outcome unequivocally demonstrates that the performance of the model is significantly influenced by both the query time and the time of former interactions.

In certain recommendation systems, the user representation vector is pre-calculated, enabling the system to utilize the pre-computed vector to generate recommendations when a query is submitted. Including query time necessitates the storage of 24 distinct user representations per user, which can increase costs, or requires running the model during inference, thereby adding complexity to the pipeline. A straightforward solution is to exclude the hour component from the query time. We conducted an experiment using the PIR-1 dataset, where the hour component of query time was removed, and the results indicated that similar improvements could be achieved with no more than a 2% degradation in the performance improvements. As an example, N@10 continued to exhibit

Table 2: The improvement in the performance of SASRec with different methods of incorporating time.

	N@1	N@10	N@20	R@1	R@10	R@20
Improv., unidimensional method (PIR-1)	13.77%	11.47%	9.68%	13.77%	7.85%	6.69%
Improv., embedding based method (PIR-1)	55.75%	30.42%	25.5%	55.75%	16.18%	9.71%
Improv., projection based method (PIR-1)	57.68%	30.50%	26.74%	57.68%	17.34%	10.82%
Results, without time (ML-20M)	0.066	0.144	0.167	0.066	0.247	0.338
Results, projection based method (ML-20M)	0.071	0.155	0.178	0.071	0.263	0.354
Improv., unidimensional method (ML-20M)	3.27%	2.56%	1.87%	3.27%	2.11%	1.39%
Improv., embedding based method (ML-20M)	4.95%	5.32%	4.59%	4.95%	5.16%	3.78%
Improv., projection based method (ML-20M)	7.38%	6.94%	6.04%	7.38%	6.29%	4.69%

a 20% improvement, while the improvement in R@10 decreased slightly from 11% to 10%.

Table 3: The improvements in the performance of Bert4Rec model with PIR-1 dataset. Scenario-I refers to including both the query time and the time of former interactions. In Scenario-II, only the time of former interactions is included and in Scenario-III, only the query time is provided to the model. N@k and R@k refers to recall and NDCG at top-k recommendations, respectively.

	N@1	N@6	R@6	N@10	R@10
Scenario-I	34.02%	22.85%	14.86%	20%	10.66%
Scenario-II	25.31%	13.87%	6.47%	11.88%	5.23%
Scenario-III	17.16%	12.3%	7.52%	9.98%	6.12%

Conclusion

Our study focused on incorporating time into sequential recommendation models, which traditionally rely on the chronological order of user activities. Rather than using time as a one-dimensional signal, we proposed to treat time as a multidimensional time-series and introduced two time-stamp encoding methods. These methods transform decomposed time-stamps into embedding vectors, which can be easily added to the recommendation, prediction, or ranking model input. Our experiments demonstrated that incorporating time improves model performance, and that the multidimensional methods outperform the one-dimensional method.

REFERENCES

[1] Alexander Dallmann, Daniel Zoller, and Andreas Hotho. 2021. A Case Study on Sampling Strategies for Evaluating Neural Sequential Item Recommendation Models. In *Fifteenth ACM Conference on Recommender Systems*. 505–514.
[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).
[3] Ziwei Fan, Zhiwei Liu, Jiawei Zhang, Yun Xiong, Lei Zheng, and Philip S Yu. 2021. Continuous-time sequential recommendation with temporal graph collaborative transformer. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 433–442.
[4] Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. 2020. A survey on knowledge graph-based recommender systems. *IEEE Transactions on Knowledge and Data Engineering* (2020).
[5] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)* 5, 4 (2015), 1–19.
[6] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*. 507–517.
[7] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of the 27th ACM international conference on information and knowledge management*. 843–852.
[8] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).
[9] Juyong Jiang, Yingtao Luo, Jae Boum Kim, Kai Zhang, and Sunghun Kim. 2021. Sequential Recommendation with Bidirectional Chronological Augmentation of Transformer. *arXiv preprint arXiv:2112.06460* (2021).
[10] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*. IEEE, 197–206.
[11] Seyed Mehran Kazemi, Rishab Goel, Sepehr Eghbali, Janahan Ramanan, Jaspreet Sahota, Sanjay Thakur, Stella Wu, Cathal Smyth, Pascal Poupart, and Marcus Brubaker. 2019. Time2vec: Learning a vector representation of time. *arXiv preprint arXiv:1907.05321* (2019).
[12] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[13] Jiacheng Li, Yujie Wang, and Julian McAuley. 2020. Time interval aware self-attention for sequential recommendation. In *Proceedings of the 13th international conference on web search and data mining*. 322–330.
[14] Aleksandr Petrov and Craig Macdonald. 2022. Effective and Efficient Training for Sequential Recommendation using Recency Sampling. In *Proceedings of the 16th ACM Conference on Recommender Systems*. 81–91.
[15] Aleksandr Petrov and Craig Macdonald. 2022. A Systematic Review and Replicability Study of BERT4Rec for Sequential Recommendation. In *Proceedings of the 16th ACM Conference on Recommender Systems*. 436–447.
[16] Ruihong Qiu, Zi Huang, Tong Chen, and Hongzhi Yin. 2021. Exploiting positional information for session-based recommendation. *ACM Transactions on Information Systems (TOIS)* 40, 2 (2021), 1–24.
[17] Ruihong Qiu, Zi Huang, Jingjing Li, and Hongzhi Yin. 2020. Exploiting cross-session information for session-based recommendation with graph neural networks. *ACM Transactions on Information Systems (TOIS)* 38, 3 (2020), 1–23.
[18] Ruihong Qiu, Hongzhi Yin, Zi Huang, and Tong Chen. 2020. Gag: Global attributed graph neural network for streaming session-based recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 669–678.
[19] Ali Rahimi and Benjamin Recht. 2007. Random features for large-scale kernel machines. *Advances in neural information processing systems* 20 (2007).
[20] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings*

of the 19th international conference on World wide web. 811–820.

[21] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*. 1441–1450.

[22] Lianghao Xia, Chao Huang, Yong Xu, and Jian Pei. 2022. Multi-Behavior Sequential Recommendation with Temporal Graph Transformer. *IEEE Transactions on Knowledge and Data Engineering* (2022).

[23] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. 2020. Inductive representation learning on temporal graphs. *arXiv preprint arXiv:2002.07962* (2020).

[24] Wenwen Ye, Shuaiqiang Wang, Xu Chen, Xuepeng Wang, Zheng Qin, and Dawei Yin. 2020. Time matters: Sequential recommendation with complex temporal information. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1459–1468.

[25] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M Jose, and Xiangnan He. 2019. A simple convolutional generative network for next item recommendation. In *Proceedings of the twelfth ACM international conference on web search and data mining*. 582–590.