Seq2Bubbles: Region-Based Embedding Learning for User Behaviors in Sequential Recommenders

Qitian Wu
Shanghai Jiao Tong University
echo740@sjtu.edu.cn

Chenxiao Yang
Shanghai Jiao Tong University
chr26195@sjtu.edu.cn

Shuodian Yu
Shanghai Jiao Tong University
timplex233@sjtu.edu.cn

Xiaofeng Gao
Shanghai Jiao Tong University
gao-xf@cs.sjtu.edu.cn

Guihai Chen
Shanghai Jiao Tong University
gchen@cs.sjtu.edu.cn

ABSTRACT

User behavior sequences contain rich information about user interests and are exploited to predict user's future clicking in sequential recommendation. Existing approaches, especially recently proposed deep learning models, often embed a sequence of clicked items into a single vector, i.e., a point in vector space, which suffer from limited expressiveness for complex distributions of user interests with multi-modality and heterogeneous concentration. In this paper, we propose a new representation model, named as *Seq2Bubbles*, for sequential user behaviors via embedding an input sequence into a set of *bubbles* each of which is represented by a center vector and a radius vector in embedding space. The bubble embedding can effectively identify and accommodate multi-modal user interests and diverse concentration levels. Furthermore, we design an efficient scheme to compute distance between a target item and the bubble embedding of a user sequence to achieve next-item recommendation. We also develop a self-supervised contrastive loss based on our bubble embeddings as an effective regularization approach. Extensive experiments on four benchmark datasets demonstrate that our bubble embedding can consistently outperform state-of-the-art sequential recommendation models.

CCS CONCEPTS

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

KEYWORDS

Sequential Recommendation, Representation Learning, User Behavior Modeling, Collaborative Filtering

ACM Reference Format:
Qitian Wu, Chenxiao Yang, Shuodian Yu, Xiaofeng Gao, and Guihai Chen. 2021. Seq2Bubbles: Region-Based Embedding Learning for User Behaviors in Sequential Recommenders. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM '21), November 1–5, 2021, Virtual Event, QLD, Australia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3459637.3482296

*Xiaofeng Gao is the corresponding author. The affiliation is MoE Key Lab of Artificial Intelligence, Department of Computer Science and Engineering, Shanghai Jiao Tong University, China.

CIKM '21, November 1–5, 2021, Virtual Event, QLD, Australia

ACM ISBN 978-1-4503-8446-9/21/11…$15.00
https://doi.org/10.1145/3459637.3482296

1 INTRODUCTION

User behavior sequences, which contain temporal records of user's interactions (click, add, purchase, etc.) or activities (like, collect, share, etc.) on one platform, often provide rich information about user profiles. In many mobile applications, such behavior sequences are widely and deeply exploited to uncover user's intentions, infer one's hidden interests and predict future behaviors in order for target recommendation. Moreover, the temporal dependencies and causal relations exposed in the sequences can further help companies to understand user behaviors and refine product designs in the future. Therefore, an effective model that can distill the rich information behind sequential user behaviors would bring up significant societal and economic benefits.

In recommender systems [11, 17], a user behavior sequence records one's historically clicked items and a common problem is to predict the next item clicked by the user based on the sequence. There are a surge of research works that propose a variety of methods to deal with the problem from different perspectives. For example, some early works [13, 15] leverage user's clicked items to represent user's interests by combining the latent factors of historically clicked items. Later, some studies [4, 6, 7, 26] build sequential models based on Markov chains and state transition to capture relations between adjacent clicked items in the sequence. Recently, deep learning models, like recurrent neural networks [9, 10, 42], memory networks [2, 12] and attention mechanisms [14, 19, 22, 30], are utilized to encode complex long-term and short-term temporal dependencies in sequences through hidden states. Such deep networks provide sufficient capacity to learn non-linear item-item relations from data and possess enough flexibility for variable-length high-dimensional sequences.

In general, the above-mentioned models inherit a seminal idea from fundamental matrix factorization model [17, 25, 27], i.e., factorizing a user-item interaction matrix into two sets of latent factors and then transforming recommendation problem into computing the distance between a user-item pair by dot-product of two factors. Based on this, as shown in Fig. 1(a), most existing sequential recommendation models can be unified as a three-step paradigm: i) *Embedding:* transform a sequence of clicked items into a sequence

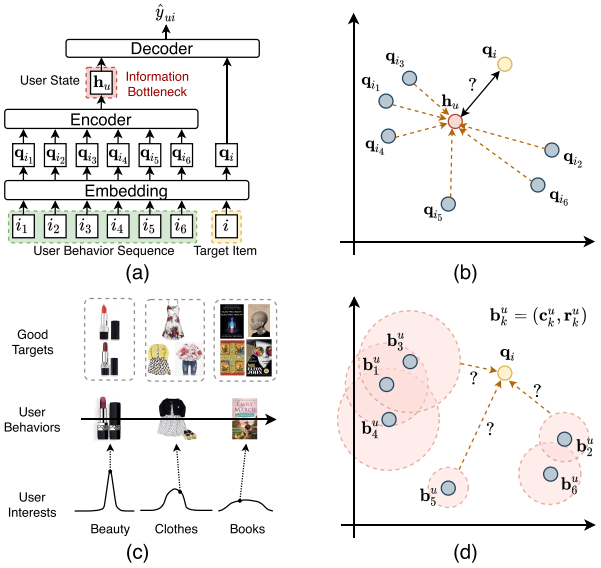


Figure 1: (a) A unified framework for most existing sequential recommendation methods. (b) The limitation of previous works that embed an input sequence into a single point in vector space. (c) A toy example of real-world user interests characterized as multi-modal distributions with heterogeneous concentration. (d) The proposed embedding approach which maps a user sequence into a set of closed regions.

of low-dimensional embedding vectors; ii) *Encoding:* encode a sequence via a sequential model (e.g., an attention network) which estimates a user state vector as representation of user interests; iii) *Decoding:* compute similarity between the user state and a target item's embedding to estimate clicking probability [1]. However, we point out that the encoding phase in above methodology suffers from an inherent limitation that the user state representation would overly squash a high-dimensional sequence into a single point in vector space (as in Fig. 1(b)). This enforces a serious information bottleneck [33] and leads to poor expressiveness for complex distributions of user interests. We provide an illustrative example in Fig. 1(c) and a thorough discussion as follows.

First, user interests often distribute over items of different aspects (like beauty, clothes, books, etc.) that form multiple clusters in physical world. A user behavior sequence often contains items of different aspects. Therefore, embedding a sequence into a single point in vector space may fail to capture such *multi-modal interest distributions*[2]. **Second**, user interests for different items have distinct concentration levels. For example, as shown in Fig. 1(c), a user loves lipsticks of a specific brand, dresses of a certain style, and books in some genres. The concentration of user interests degrades from the first case to the last. Formally speaking, we can define user's concentration in one aspect as variance of user's clicked items in such aspect and users would like more (resp. less) diverse items in the aspect with stronger (resp. weaker) concentration. Naturally, the concentration is often distinct over different aspects for one user. Nevertheless, existing methods would fail to capture such a characteristic since the point embedding implicitly assumes that user's concentration is the same among different aspects. In fact, we note that the distinct concentration levels can be essentially identified from other clicked items in the sequence (e.g., whether she clicked lipsticks of other brands).

In this paper, we propose a novel representation model, named as *Seq2Bubbles*, for user behaviors in sequential recommendation. We embed a sequence into a set of bubbles (formally, each of which is a closed region inside a hyper-ellipsoid) instead of a point in vector space. In specific, as illustrated in Fig. 1(d), each clicked item in the sequence will be first mapped to a center vector in the vector space using one-hot embeddings and then we estimate a radius vector for each center using the sequence of historically clicked items. To maintain enough representation capacity, we design a special hierarchical encoder which contains a low-level sequential unit and a high-level readout unit. The sequential unit can filter out noise in an input sequence and capture temporal dependencies, while the readout unit adaptively aggregates historical information to deduce concentration of user interests. The bubble embedding possesses desirable expressiveness for multi-modal distributions as well as distinct concentration of user interests.

We also develop an efficient scheme to compute distance/similarity between a target item and the bubble embedding to achieve next-item recommendation. In particular, to alleviate over-fitting, we devise a self-supervised contrastive learning loss based on our bubble embeddings as an effective regularization scheme. We conduct comprehensive experiments on four benchmark datasets for sequential recommendation and compare with several strong competitors (e.g., [19, 22, 30]). The results show that our model yields significant improvement w.r.t. Hit Ratios and NDCGs. We summarize our contributions as follows:

i) **Methodology:** We propose a new representation model for sequential user behaviors in recommendation. The method embeds a sequence into a set of closed regions in vector space and possesses superior expressiveness, especially for complex distributions of user interests with multi-modality and heterogeneous concentration.

ii) **Techniques:** We develop effective techniques to enhance the practicality of our model. We design an efficient distance/similarity computing scheme for our new embedding approach to achieve next-item recommendation. Also, we devise a self-supervised contrastive loss as regularization to enhance model learning.

iii) **Evaluation:** We conduct extensive experiments and compare with various competitors. The results demonstrate that our model achieves state-of-the-art performance on several sequential recommendation benchmarks. Our comprehensive ablation studies also thoroughly dissect the effectiveness of proposed modules.

2 PRELIMINARY

We consider a set of M users denoted as $\mathcal{U}$ and their interaction behaviors over a set of N items denoted as $\mathcal{I}$. Define $\mathcal{T}_u = \{i_1^u, i_2^u, \cdots, i_{n_u}^u\}$ as user u's behavior sequence, where i_m^u is the m-th clicked item by user u (in chronological order), and $\mathcal{T}_u^t =$

[1] Some models also concatenate the user state with user embedding or context feature embedding in the decoding layer.

[2] The *multi-modal distribution* means a distribution with multiple modes. It is different from multi-modal data which means data from multi-media sources (such as vision, texts, audio, etc.) in deep learning community.

$\{i_1^u, i_2^u, \cdots, i_t^u\}$ as a sequence of user u's clicked items up to i_t^u. Our goal is to predict the next item that a user may click based on the historical behavior sequence, which can be formulated as below.

Problem Formulation: given a set of user behavior sequences $\{\mathcal{T}_u\}_{u \in \mathcal{U}}$ as training data, we aim to model the conditional probability $P(i_t|\mathcal{T}_u)$ that measures how likely a user u would click target item i_t based on the behavior sequence $\mathcal{T}_u$.

3 PROPOSED MODEL

In this section, we present our proposed model *Seq2Bubbles*, a new embedding approach for user behavior sequences in sequential recommendation. In Fig. 2 we show our model framework which takes a user sequence $\mathcal{T}_u$ as input and estimates a *bubble embedding* as its representation. Based on this, the recommendation problem boils down to computing the similarity between a bubble embedding and a target item's embedding (i.e., a point in vector space).

In Section 3.1, we introduce our region-based embedding model where we formally define bubble embeddings, discuss how to compute the bubble embedding and present an efficient approach for calculating similarity between a point and a bubble embedding in vector space. In Section 3.2, we introduce context-aware bubble embedding as a supplement to our model. In Section 3.3, we focus on model optimization where we first give supervised learning objective and devise a self-supervised contrastive loss as an effective regularization for training.

3.1 Region-based Embedding Model

We consider a behavior sequence $\mathcal{T}_u^m = \{i_1^u, i_2^u, \cdots, i_m^u\}$ for user u as input. To avoid redundancy, we drop the subscript for $\mathcal{T}_u^m$ and consider a sequence $\mathcal{T}^m = \{i_1, i_2, \cdots, i_m\}$ for presentation in this subsection.

Item Embedding Layer. Following existing methods, we first map items to low-dimensional embeddings using an embedding matrix $\mathbf{H} \in \mathbb{R}^{N \times d}$ where d is embedding size. Item i can be denoted as a one-hot column vector $\mathbf{e}_i \in \{0,1\}^N$, where $\mathbf{e}_i(k) = 1$ and $\mathbf{e}_i(k') = 1$, for $k' \neq k$, if i is the k-th item in $\mathcal{I}$. Then the embedding vector for item i can be given by $\mathbf{q}_i = \mathbf{e}_i^\top \mathbf{H}$. The parameters of $\mathbf{H}$ will be learned together with the model through end-to-end learning. Based on the item embedding, we transform the sequence $\mathcal{T}^m$ into a sequence of item embeddings $\mathcal{Q}^m = \{\mathbf{q}_{i_1}, \mathbf{q}_{i_2}, \cdots, \mathbf{q}_{i_m}\}$ as a basis for subsequent model.

Bubble Embedding Definition. We define a bubble in vector space as $\mathbf{b} = (\mathbf{c}, \mathbf{r})$ where $\mathbf{c} \in \mathbb{R}^d$ denotes a center vector and $\mathbf{r} \in \mathbb{R}_+^d$ denotes a radius vector. A bubble embedding represents a closed region $\{\mathbf{x} : \|(\mathbf{x} - \mathbf{c}) \odot \frac{1}{\mathbf{r}}\|_2 \leq 1, \mathbf{x} \in \mathbb{R}^d\}$, i.e. a hyper-ellipsoid, in vector space, where $\odot$ denotes element-wise product. For sequence $\mathcal{T}^m = \{i_1, i_2, \cdots, i_m\}$, we transform it into a set of bubbles $\mathcal{B}^m = \{\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_m\}$ where $\mathbf{b}_k = (\mathbf{c}_k, \mathbf{r}_k)$. Each bubble $\mathbf{b}_k$ encloses a region (in vector space) that contains an infinite set of points some of which correspond to existing items in candidate set $\mathcal{I}$ [3]. Naturally, the items inside $\mathbf{b}_k$ should have similar embeddings to the clicked item i_k and can be treated as the ones that the user may click. Furthermore, the bubble embedding $\mathcal{B}^m$ for sequence $\mathcal{T}^m$ defines a union of (disjoint or intersected) regions in vector space, which can be denoted as an area:

$$\bigcup_{k=1}^{m} \{\mathbf{x} : \|(\mathbf{x} - \mathbf{c}_k) \odot \frac{1}{\mathbf{r}_k}\|_2 \leq 1\}. \tag{1}$$

Based on our definition and reasoning, such area can be seen as an estimation of a finite set of items that the user would click in the future based on the history sequence $\mathcal{T}^m$. Therefore, it can be a representation for user interests reflected by $\mathcal{T}^m$.

The bubble embedding has several merits: i) *Superior Expressiveness:* The (different) center vectors for clicked items accommodate multi-modality in user interests, and the radius vectors can represent distinct concentration levels of user interests in each aspect. ii) *Enough Flexibility:* The bubble embedding maintains the flexibility for handling variable-length high-dimensional input sequences. It requires no specific domain knowledge about the input sequences. iii) *Interpretability*: The regions enclosed by bubbles can explicitly reflect user's interests distributed over the latent space where the distance to item points reflect estimated user preference on these items. Such representation can be further used to interpret user intentions and construct user profiles for various downstream tasks.

Encoding Layer (Bubble Embedding Estimation). We next present how to compute each bubble $\mathbf{b} = (\mathbf{c}, \mathbf{r})$ to make it work in practice. Since a bubble denotes a region that encloses a set of items related to the clicked one, we can use the item embedding as bubble's center $\mathbf{c}_k = \mathbf{q}_{i_k}$. Moreover, the radius for bubbles aims to embody concentration of user interests. Presumably, the concentration characteristic can be identified with the observed clicked items in the sequence. Concretely, if a user clicked several items that have very similar embeddings within a small region (i.e., a user may like specific products of a specific brand), the concentration in such aspect could be strong. On the other hand, if a user clicked items whose embeddings are not very close to each other (i.e., a user likes products of one style), the concentration in this aspect would be relatively weak. To enable the bubble embedding to capture such characteristics, we resort to deep network's representation capacity and consider an encoder model $\Phi(\cdot)$ to leverage the sequence of item embeddings and estimate the radius:

$$[\mathbf{r}_1, \mathbf{r}_2, \cdots, \mathbf{r}_m] = \Phi(\mathcal{Q}^m). \tag{2}$$

With the aim of extracting useful information from input sequence, the encoder $\Phi(\cdot)$ needs to meet three principles. First, it needs to filter out noise existing in behavior sequences which may contain mistakenly clicking by users. Second, it should be able to encode temporal dependency among behaviors since user's interests may evolve over time. Third, it needs to distinguish the importance of different historical behaviors according to different states. To these ends, we design a hierarchical architecture for $\Phi(\cdot)$ which contains a low-level sequential unit $\Phi_A(\cdot)$ and a high-level readout unit $\Phi_R(\cdot)$.

The sequential unit takes $\mathcal{Q}^m$ as input and outputs a sequence of hidden states with equal length:

$$[\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_m] = \Phi_A(\mathcal{Q}^m). \tag{3}$$

We specify $\Phi_A(\cdot)$ as a self-attention network [36] and adopt the self-attentive architecture by [14] for its advanced capacity. For

[3]The item embedding layer defines a one-to-one mapping from a finite set of items in $\mathcal{I}$ to a series of points in a continuous latent space which essentially contains an infinite and uncountable set of points.

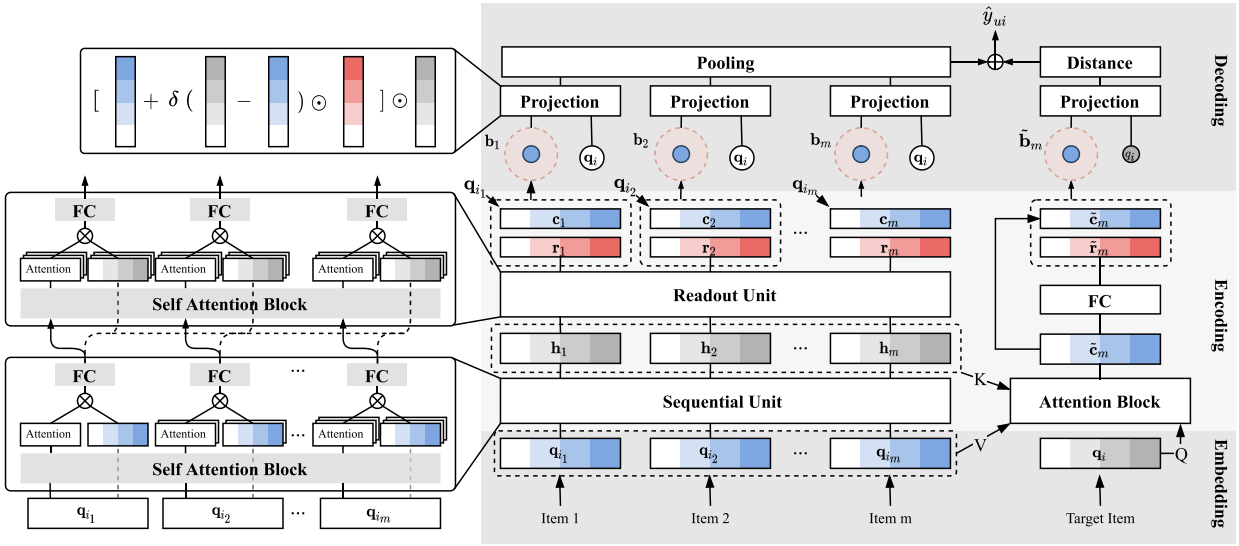


Figure 2: Illustration of proposed model Seq2Bubbles which contains three layers: embedding, encoding and decoding. In embedding layer, each item in $\mathcal{T}_u$ is mapped into low-dimensional embeddings. In encoding layer, such item embeddings are used to compute a *bubble embedding* which consists of a set of bubbles. Each bubble is represented by a center vector and radius vector, and defines a closed region in vector space. The union of these regions encloses an area that contains a set of potential items (whose embeddings are located inside the area) that could be clicked by the user. Hence, such bubble embedding can be an effective representation for user's interests reflected by the sequence. In decoding layer, a similarity score between the bubble embedding and the target item's embedding is estimated in order for next-item recommendation.

input sequence $Q^m = [\mathbf{q}_{i_1}, \cdots, \mathbf{q}_{i_m}]$, we define (for $k = 1, \cdots, m$)

$$\mathbf{z}_k = \sum_{j=1}^{k} \alpha_{jk} \mathbf{q}_{i_j}, \quad \text{where } \alpha_{jk} = \sigma\left(\frac{(\mathbf{W}_K^1 \mathbf{q}_{i_k})^\top (\mathbf{W}_Q^1 \mathbf{q}_{i_j})}{\sqrt{d}}\right), \tag{4}$$

where σ is a sigmoid function, $\mathbf{W}_K \in \mathbb{R}^{d \times d}$, $\mathbf{W}_Q \in \mathbb{R}^{d \times d}$ are weight matrices. Here we do not use softmax function for normalization over outputs as original self-attention network [36]. Instead, the unnormalized attention score $\alpha_{jk} \in (0, 1)$ can help to filter out noise and unrelated information in input sequences. Similar design is also used by [44]. We proceed to stack a neural layer to model deeper feature-wise interaction (for $k = 1, \cdots, m$):

$$\mathbf{h}_k = Dropout(PReLU(\mathbf{W}_N^1 \mathbf{z}_k + \mathbf{b}_N^1)), \tag{5}$$

where $\mathbf{W}_N^1 \in \mathbb{R}^{d \times d}$ is a weight matrix and $\mathbf{b}_N^1 \in \mathbb{R}^d$ is a bias vector. Here we use *PReLU* as non-linear activation and *Dropout* for enhancing generalization. The sequential unit accommodates causal direction in the chronological order of user behaviors and can encode temporal dependencies among items. Moreover, the attention score α_{jk} aggregates historical items in a weighted manner in order to attach different importance according to user dynamic states.

We proceed to consider the readout unit $\Phi_R(\cdot)$ that leverages hidden states to estimate radius of bubbles:

$$[\mathbf{r}_1, \mathbf{r}_2, \cdots, \mathbf{r}_m] = \Phi_R([\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_m]). \tag{6}$$

The readout unit needs to aggregate similar behaviors in user's historically clicked items and model the concentration of user interests in different aspects. To this end, we again adopt self-attention network. Differently, here we consider attention over a whole sequence instead of historical items. Concretely, we have (for $k = 1, \cdots, m$)

$$\mathbf{z}_k = \sum_{j=1}^{m} \beta_{jk} \cdot \mathbf{h}_j, \quad \text{where } \beta_{jk} = \sigma\left(\frac{(\mathbf{W}_K^2 \mathbf{h}_k)^\top (\mathbf{W}_Q^2 \mathbf{h}_j)}{\sqrt{d}}\right), \tag{7}$$

where $\mathbf{W}_K^2 \in \mathbb{R}^{d \times d}$, $\mathbf{W}_Q^2 \in \mathbb{R}^{d \times d}$ are weight matrices. Then we adopt a neural layer to obtain the radius of bubble:

$$\mathbf{r}_k = Softplus(\mathbf{W}_N^2 \mathbf{z}_k + \mathbf{b}_N^2), \quad k = 1, \cdots, m. \tag{8}$$

where $\mathbf{W}_N^2 \in \mathbb{R}^{d \times d}$, $\mathbf{b}_N^2 \in \mathbb{R}^d$ and we use a softplus function to enforce positive values for each value of $\mathbf{r}_k$. We now obtain the bubble embedding $\mathcal{B}^m = \{\mathbf{b}_k\}_{k=1}^m = \{(\mathbf{c}_k, \mathbf{r}_k)\}_{k=1}^m$.

Decoding Layer (Distance/Similarity Computation). The next-item recommendation problem is to predict whether a user would click a target item. As discussed above, the bubble embedding reflects user interests by defining an area in the vector space. Existing sequential recommendation models tend to transform recommendation into estimating the similarity between user state vector (i.e., output of a sequential model) and a target item's embedding. Therefore, similarly, we can convert the problem into computing the similarity between the bubble embedding of a sequence and a target item's embedding (i.e., a point in vector space). Since similarity measures are usually in some sense the inverse of distance metrics, we now discuss how to calculate the distance between $\mathcal{B}^m$ and a target item i.

Note that it is intractable to exactly compute the distance from a point to the surface of a hyper-ellipsoid in vector space. Therefore, we resort to an approximation approach. We consider a circumscribed hyper-cube outside the hyper-ellipsoid region defined by a

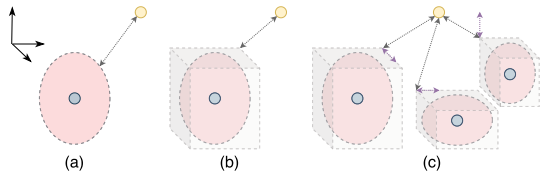


Figure 3: An illustration of proposed distance/similarity computation. (a) Original distance from a point to a bubble. (b) Approximated distance from a point to the closest vertex of a circumscribed hyper-cube. (c) Generalized distance/similarity measure using max-pooling to select dominant bubbles in each feature dimension.

bubble $\mathbf{b} = \{\mathbf{c}, \mathbf{r}\}$: $[c_1 - r_1, c_1 + r_1] \times \cdots \times [c_d - r_d, c_d + r_d]$, where c_l (resp. r_l) is the l-th value of $\mathbf{c}$ (resp. $\mathbf{r}$). The hyper-cube has 2^d vertices denoted as $\{\mathbf{c}+\mathbf{e}\odot\mathbf{r} : \mathbf{e} \in \{-1, 1\}^d\}$ where $\mathbf{e}$ is a d-dimensional binary vector whose values are from $\{-1, 1\}$. We then focus on the distance from the target point to the closest vertex of the hyper-cube. Fig. 3(a) and (b) present an illustration for the approximation. Formally, the distance between a bubble and a point is defined as

$$D(\mathbf{b}, \mathbf{q}) \coloneqq \min_{\mathbf{e}\in\{-1,1\}^d} d(\mathbf{c} + \mathbf{e} \odot \mathbf{r}, \mathbf{q}), \tag{9}$$

where $d(\cdot, \cdot)$ is a distance measure in vector space, like Euclidean distance.

Note that (9) can be computed in an efficient alternative way.

Theorem 3.1. *The distance measure in* (9) *can be equivalently written as:*

$$D(\mathbf{b}, \mathbf{q}) = d(\mathbf{c} + \delta(\mathbf{q} - \mathbf{c}) \odot \mathbf{r}, \mathbf{q}), \tag{10}$$

where $\delta : \mathbf{R}^d \to \{-1, 1\}^d$ is an element-wise indicator function (the l-th value of $\delta(\mathbf{x})$ equals to 1 if $x_l > 0$ and otherwise, equals to -1).

Proof. We prove the equivalence between (9) and (10) in Section 3.1. Consider a bubble $\mathbf{b} = \{\mathbf{c}, \mathbf{r}\}$. First, we divide the hyper-plane $\mathbb{R}^d$ into 2^d different areas according to the indicator function $\delta(\mathbf{q} - \mathbf{c}) \in \{-1, 1\}^d$. For area A_n, we define $A_n = \{\mathbf{q} | \delta(\mathbf{q} - \mathbf{c}) = \mathbf{e}_n\}$, $n = 1, \cdots, 2^d$, where $\mathbf{e}_n$ is a d-dimensional binary vector whose entries are from $\{-1, 1\}$. Note that for any point $\mathbf{q} \in A_n$, $\delta(\mathbf{q} - \mathbf{c})$ is the same and A_n contains one and only one vertex of the circumscribed hyper-cube, denoted as $\mathbf{v}_n = \mathbf{c} + \mathbf{e}_n \odot \mathbf{r} \in A_n$. Consider a point $\mathbf{q}' \in A_n$. If $\mathbf{q}'$ is outside the bubble $\mathbf{b}$, we make a line from $\mathbf{q}'$ to $\mathbf{c}$ which will intersect with the surface of the bubble at a point $\mathbf{x}$. Obviously we have $\mathbf{x} \in A_n$ and $\delta(\mathbf{q}' - \mathbf{c}) = \delta(\mathbf{x} - \mathbf{c})$. We know that vertex $\mathbf{v}_n$ is the closest one to $\mathbf{q}'$. Otherwise, if there exists $\mathbf{v}_{n'}$ such that $d(\mathbf{v}_{n'}, \mathbf{q}') < d(\mathbf{v}, \mathbf{q}')$, then we have $d(\mathbf{v}_{n'}, \mathbf{x}) < d(\mathbf{v}, \mathbf{x})$, which results in contradictory. If $\mathbf{q}'$ is outside $\mathbf{b}$, we can extend the line from $\mathbf{c}$ to $\mathbf{q}'$ which will intersect with the surface of $\mathbf{b}$ at a point $\mathbf{x}$. Then we can finish the proof in a similar way. We present an illustration for $d = 2$ in Fig. 4. □

The equivalence from (9) to (10) can significantly reduces the time complexity for distance computation from $O(2^d)$ to $O(d)$.

We can further define the distance between bubble embedding $\mathcal{B}^m$ and target item i as the minimum of distances to each bubble

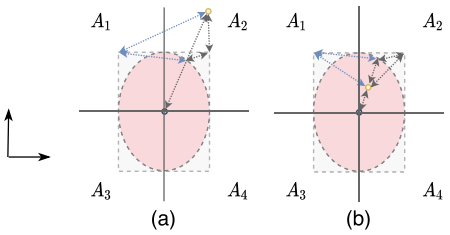


Figure 4: An illustration of proof when $d = 2$.

in the sequence

$$\begin{aligned}\mathcal{D}(\mathcal{B}^m, \mathbf{q}_i) &\coloneqq \min_{1\le k\le m} D(\mathbf{b}_k, \mathbf{q}_i), \\ &= \min_{1\le k\le m} d(\mathbf{c}_k + \delta(\mathbf{q}_i - \mathbf{c}_k) \odot \mathbf{r}_k, \mathbf{q}_i).\end{aligned} \tag{11}$$

Correspondingly, the similarity score between bubble embedding $\mathcal{B}^m$ and item i can be defined by

$$\mathcal{S}(\mathcal{B}^m, \mathbf{q}_i) = \max_{1\le k\le m} s(\mathbf{c}_k + \delta(\mathbf{q}_i - \mathbf{c}_k) \odot \mathbf{r}_k, \mathbf{q}_i), \tag{12}$$

where $s(\cdot, \cdot)$ is a similarity measure.

However, there are two drawbacks for the above similarity (and distance) definition. First, the maximum operation only selects one bubble in the sequence and the gradient for optimization will only update one item in the sequence at a time, which leads to slow model learning and instability. Second, the maximum operation is taken over m scalars given by $s(\cdot, \cdot)$ and will ignore effects from different feature dimensions. To address these issues, we propose a generalized version of similarity measure for bubble embeddings by first considering m projection functions and taking a max-pooling along the feature dimension:

$$\begin{aligned}&\mathbf{p}_k = [\mathbf{c}_k + \delta(\mathbf{q}_i - \mathbf{c}_k) \odot \mathbf{r}_k] \odot \mathbf{q}_i, \quad k = 1, \cdots, m, \\ &\mathbf{a}_m = \text{MaxPooling}\{[\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_m]\} \\ &\mathcal{S}(\mathcal{B}_m, \mathbf{q}_i) = s(\mathbf{a}_m, \mathbf{q}_i).\end{aligned} \tag{13}$$

An illustration for the computation of the generalized similarity measure is shown in Fig. 3(c).

In sequential recommendation methods [14, 19, 22, 30], a widely adopted similarity measure is dot-product. Here, we also specify s as dot-product and the relevance score of user u and item i based on sequence $\mathcal{T}^m$ can be estimated by:

$$\hat{y}^m_{ui} = (\mathbf{q}_i)^\top \mathbf{a}_m. \tag{14}$$

The time complexity for computing the similarity is $O(md)$, which is comparable as previous models that also requires at least $O(md)$ to obtain the state representation of a sequence and then compute the similarity between the state vector and target item's embedding. The total feed-forward time complexity for our region-based embedding model is $O(m^2d+md^2)$ (induced by self-attention networks and fully-connected networks), the same as attentive sequential recommendation models [14, 19].

3.2 Context-Aware Bubble Embedding

The bubble embedding $\mathcal{B}^m$ models a distribution of user interests in vector space given observed history sequence $\mathcal{T}^m$. However, $\mathcal{B}^m$ is independent of target items and may be inadequate for modeling relations between user's clicked items and the target one. As a

supplement for our model, we incorporate a context-aware state representation (also as a bubble) $\tilde{\mathbf{b}}_m = [\tilde{\mathbf{c}}_m, \tilde{\mathbf{r}}_m]$ to encode information related to the target item. For target item i, we use a weighted sum of clicked items in the sequence as estimation for bubble's center by another attention network:

$$\tilde{\mathbf{c}}_m = \sum_{k=1}^{m} \gamma_{km} \mathbf{q}_{i_k}, \quad \text{where } \gamma_{km} = \sigma\left(\frac{(\mathbf{W}_K^3 \mathbf{h}_k)^\top (\mathbf{W}_Q^3 \mathbf{q}_i)}{\sqrt{d}}\right), \quad (15)$$

where $\mathbf{h}_k$'s are the hidden states given by the sequential unit in (3). The bubble's radius can be given as

$$\tilde{\mathbf{r}}_u^m = Softplus(\mathbf{W}_N^3 [\tilde{\mathbf{c}}_m \| \mathbf{q}_i] + \mathbf{b}_N^3), \quad (16)$$

where $\mathbf{W}_N^3 \in \mathbb{R}^{2d \times d}$ and $\mathbf{b}_N^3 \in \mathbb{R}^d$. The distance between $\tilde{\mathbf{b}}_m$ and $\mathbf{q}_i$ can also be calculated by (10). Therefore, we can incorporate such context-aware bubble into (14) and obtain the estimated relevance score as

$$\hat{y}_{ui}^m = (\mathbf{q}_{i_t})^\top \mathbf{a}_m + (\mathbf{q}_{i_t})^\top \tilde{\mathbf{p}}_m, \quad (17)$$

where $\tilde{\mathbf{p}}_m = \tilde{\mathbf{c}}_m + \delta(\mathbf{q}_i - \tilde{c}_m) \odot \tilde{\mathbf{r}}_m$. The bubble embedding $\mathcal{B}^m$ and the context-aware state $\tilde{\mathbf{b}}_m$ can work collaboratively to learn effective user representations by modeling 1) user's inherent interests from observed sequence and 2) relations between user's historical behaviors and target items, respectively.

3.3 Model Optimization

Based on our bubble embedding model, we obtain the predicted relevance score $\hat{y}_{ui}^m$ for target item i based on history sequence $\mathcal{T}_u^m$ (where we add the subscript u for remaining presentation). In the following, we first give supervised learning objective using ground-truth sequences $\{\mathcal{T}_u\}$ and then propose a contrastive learning loss based on our bubble embeddings as an effective regularization scheme.

Supervised Learning. With the relevance score $\hat{y}_{ui}^m$, we can estimate the probability $P(i|\mathcal{T}_u^m) = \sigma(\hat{y}_{ui}^m)$ where σ is a sigmoid function. We adopt Bayesian Personalized Ranking (BPR) loss [25] as our objective. For each observed sequence $\mathcal{T}_u$ of user u, we sample a sequence of negative items $\overline{\mathcal{S}}_u = \{\bar{i}_1^u, \bar{i}_2^u, \cdots, \bar{i}_{n_u}^u\}$ where $\bar{i}_m^u$ is sampled from $\mathcal{I} \setminus \mathcal{T}_u^m$. Then the BPR objective can be written as

$$\mathcal{L} = \sum_{u \in \mathcal{U}} \sum_{m=1}^{n_u - 1} \log P(i_{m+1}^u \succcurlyeq \bar{i}_{m+1}^u | \mathcal{T}_u^m), \quad (18)$$

where $i_1^u \succcurlyeq i_2^u$ denotes that item i_1^u appears in front of item i_2^u in the recommendation list of user u. In practice, we adopt mini-batch training and the final objective for a batch of data $\{\mathcal{T}_u\}_{u \in \mathcal{U}_b}$ (where $\mathcal{U}_b$ denotes a batch of users sampled from $\mathcal{U}$) can be

$$\mathcal{L}_{sup} = \sum_{u \in \mathcal{U}_b} \sum_{m=1}^{n_u - 1} \log \sigma(\hat{y}_{u,i_{m+1}^u}^m - \hat{y}_{u,\bar{i}_{m+1}^u}^m). \quad (19)$$

Contrastive Regularization. Empirically, we found directly optimizing (19) would be prone for over-fitting since the radius vectors for bubbles tend to be updated in a radical manner. To address the issue, we further propose a regularization scheme by enforcing self-consistency. Inspired by contrastive learning [1, 34, 35], which has shown great power as a self-supervised learning approach for general unsupervised representation learning tasks, we introduce an auxiliary contrastive loss based on our bubble embeddings. Specifically, in terms of bubble embedding $\mathcal{B}_u^m$ for user sequence $\mathcal{T}_u^m$, we mask the t-th latest bubble and define $\overline{\mathcal{B}}_u^m = \mathcal{B}_u^m \setminus \{\mathbf{b}_u^{m-t}\}$. Then we aim at maximizing the similarity between $\overline{\mathcal{B}}_u^m$ and the historically clicked item i_{m-t}^u in the sequence and minimizing the similarity between $\overline{\mathcal{B}}_u^m$ and $i_{m-t}^{u'}$ for user u' from $\mathcal{U}_b \setminus \{u\}$. The objective is

$$\mathcal{L}_{reg} = -\sum_{u \in \mathcal{U}_b} \sum_{m=t+1}^{n_u} \log \frac{\exp(\mathcal{S}(\overline{\mathcal{B}}_u^m, \mathbf{q}_{i_{m-t}^u}))}{\sum_{u' \in \mathcal{U}_b} \exp(\mathcal{S}(\overline{\mathcal{B}}_u^m, \mathbf{q}_{i_{m-t}^{u'}}))}, \quad (20)$$

where $\mathcal{S}(\cdot, \cdot)$ is given by (13). Such contrastive loss enforces self-consistency within a user sequence in the context of other sequences in a mini-batch, which essentially enlarges the mutual information between estimated bubble embedding and clicked items in the sequence. Intuitively, the regularization can prevent the model from merely 'looking ahead' and over-fitting the next-item prediction objective (19), and meanwhile guide the model to 'look back' and fully preserve the information in observed sequences. In practice, we found $t = 1$ is enough for providing effective regularization in our experiments. The final objective is

$$\mathcal{L} = \mathcal{L}_{sup} + \lambda \mathcal{L}_{reg}, \quad (21)$$

where λ is a hyper-parameter that controls the balance between model fitting and regularization.

4 LINKS TO RELATED WORKS

We briefly discuss several lines of works related to ours.

Sequential Recommendation. Sequential recommendation seeks to leverage a sequence of user's clicked items, infer user's hidden interests, and predict the next item that would be clicked. Early studies [4–7, 16, 26] adopts Markov chains and aim to extract relations between adjacent clicked items in the sequence. Later, deep learning models are extensively adopted to deal with this problem from different perspectives [2, 9, 10, 12, 14, 18, 19, 21, 39, 42]. Some recent works also propose self-supervised learning approaches to enhance the performance based on deep learning models [22, 30]. The deep learning based methods can effectively model non-linear relations in user sequences. However, they embed an input sequence into a single point in vector space which would limit model expressiveness. Our Seq2Bubbles can essentially address this problem and also incorporate with off-the-shelf deep learning models (as our encoding model) to achieve advanced representation capacity.

User Behaviors Modeling. User behavior modeling is a general and fundamental problem in broad areas of knowledge discovery and data mining. In online advertising, CTR prediction is an important task and user behavior modeling can help to accurately predict whether a user will click an advertisement [20, 43, 44] Furthermore, in social influence analysis, user behavior modeling is also important for uncovering user-to-user associations in dynamic social networks [29, 40]. An effective model for user behavior analysis can also benefit information diffusion prediction [28] and knowledge-aware content recommendation [38]. While our paper focuses on user behavior modeling in the context of sequential recommendation, the proposed bubble embedding approach can generalize to other areas that require representation learning for observed behavior sequences and help to better interpret user interests and intentions with superior expressiveness and flexibility.

Geometrically-Inspired Embedding. Learning geometrically-inspired embeddings is of growing interests in deep learning community. [3] is one of the early works that propose region-based embedding approach for word representation. Recent works propose various embedding approaches, including box embeddings [37], spherical embeddings [24], structured embeddings [41] and complex embeddings [31], with various applications to computer vision, natural language processing, knowledge representation and reasoning, etc. Our Seq2Bubbles share similar spirits with them. User behavior sequences are essentially transitive relational data which contain unobserved item-item relations and hidden user intentions. To model such complex structures behind data, we propose bubble embeddings to represent user sequences as a well-defined closed regions in vector space, which can accommodate multi-modal distributions of user interests with diverse concentration levels.

5 EXPERIMENTS

In this section, we apply our model to several benchmark datasets for sequential recommendation and comprehensively evaluate our model by answering the following research questions:

- RQ1: How does Seq2Bubbles perform compared with state-of-the-art sequential recommendation models?
- RQ2: Are the proposed units in Seq2Bubbles effective and necessary for satisfactory performance?
- RQ3: How is the robustness of Seq2Bubbles w.r.t. noisy input sequences?
- RQ4: How would the hyper-parameters in SeqBubbles impact its performance?
- RQ5: How is the scalability of Seq2Bubbles w.r.t. sequence length and embedding dimension?

5.1 Experiment Setup

Dataset. We conduct experiments on four public datasets: Amazon-Beauty, Steam, Movielens-1M and Movielens-20M. Amazon recommendation datasets contain a series of datasets collected by [23] from Amazon platform. There are 24 categories including clothes, books, games, etc. We adopt the dataset *beauty* for evaluation. *Steam* is a dataset crawled from Steam website, a large online video game distribution platform, by [14]. *Movielens* is a widely used benchmark dataset for recommendation evaluation. We use two versions, *MovieLens-1M (ML-1M)* and *MovieLens-20M (ML-20M)* in our experiments. Following the common settings in sequential recommendation [14, 19, 22, 26, 30], for each dataset, we use all the user-item rating records as implicit feedbacks and construct a sequence of clicked items for each user in the order of timestamps. We filter out users and items with less than five interactions. For each user, we use the most recent clicked item for testing, the second recent one for validation and the remaining clicked items for training.

Evaluation Metrics. To evaluate model performance for recommendation, we adopt two widely used top-n metrics, Hit Ratio (HR) and Normalized Discounted Cumulative Gain (NDCG). Also, following [22, 30], we randomly sample 100 negative items, according to item popularity, for each user u and rank them with the ground-truth clicked item. The problem is to rank 101 items for each user based on the observed sequence. We run each method in five independent experiments and report the average value for each metric.

Compared Methods. We compare Seq2Bubbles with several baselines, including heuristic popularity-based method POP, matrix factorization models BPR-MF [25], NCF [8] and FPMC [26], sequential neural network-based models GRU4Rec [10], GRU4Rec+ [10] and Caser [32], self attention-based models SASRec [14], TiSASRec [19] and self-supervised models BERT4Rec [30], DisenRec [22]. Here, SASRec, TiSASRec, BERT4Rec and DisenRec are state-of-the-arts for sequential recommendation. We provide a brief introduction for them as follows.

POP ranks all the items by their popularity calculated by the number of interactions in the training sets. BPR-MF is a classic matrix factorization model trained with Bayesian Personalized Ranking (BPR) loss. NCF extends the matrix factorization (MF) model with multi-layer perceptron (MLP) which replaces the inner product in MF. FPMC combines MF with first-order Markov Chains to capture user's long-term preference. GRU4Rec uses Gated Recurrent Unit (GRU) to model user action sequences for session-based recommendation. GRU4Rec+ is a variant of GRU4Rec that designs an improved loss function and sampling method. SASRec captures user's sequential behaviors with a left-to-right Transformer model. TiSASRec is an improved version of SASRec that consider the time intervals between items. BERT4Rec adopts self-supervised learning to pretrain a sequential recommendation model with bidirectional self-attention mechanism. DisenRec considers disentanglement of multiple user intentions in sequences for self-supervised learning.

Implementation Details. We implement Seq2Bubbles with PyTorch. All the parameters are initialized with default initialization. We train the model via Adam optimizer with weight decay for model parameters. The maximum sequence length is set following [30]. We use validation dataset to select hyper-parameters. We consider maximum sequence length $n = 200$ for ML-1M and ML-20M, $n = 50$ for Steam and Amazon-Beauty. The optimizer is Adam with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The learning rate is 0.01 for Amazon-Beauty and 0.001 for other three datasets. The weight decay coefficient is set as 0.005 for Steam and 0.001 for ML-1M, ML-20M and Amazon-Beauty. The dropout probability is set as 0.5 and the regularization weight *lambda* is 0.0001 for all the datasets. The embedding size is set as 32 for ML-1M and ML-20M and 20 for Steam and Amazon-Beauty. Besides, the t in contrastive regularization loss is set as 1 for all the datasets. The batch size is set as 256. For comparative methods, we refer to the hyper-parameter settings in their papers and also finetune them in different datasets.

5.2 Comparative Results

We report experiment results of Seq2Bubbles and other comparative methods in Table 1. As we can see, our model Seq2Bubbles consistently outperforms all the competitors and achieve state-of-the-art results w.r.t. different metrics throughout four datasets. Concretely, compared with the best competitor, Seq2Bubbles achieves $8.4\% - 13.7\%$ improvements of NDCG@10 and $2.8\% - 7.3\%$ improvements of HR@10, which demonstrates the superior power of Seq2Bubbles for sequential recommendation. Furthermore, we find that the relative improvement achieved by Seq2Bubbles on NDCG is much larger than that on HR. It is possibly because the bubble

Table 1: Experiment results of different methods. The bold scores mark the best in each row, while the underlined scores correspond to the second best. Improvements over competitors are statistically significant with $p < 0.05$.

Datasets	Metric	POP	BPR-MF	NCF	FPMC	GRURec	GRURec+	Caser	SASRec	TiSASRec	BERT4Rec	DisenRec	Seq2Bubbles	Improv.
Beauty	N@5	0.0241	0.0803	0.0844	0.0921	0.0821	0.1186	0.1054	0.1439	0.1310	0.1585	0.2404	**0.2767**	+13.1%
	H@5	0.0396	0.1219	0.1304	0.1372	0.1321	0.1791	0.1613	0.1929	0.1804	0.2201	0.3225	**0.3508**	+8.0%
	N@10	0.0337	0.1059	0.1132	0.1215	0.1064	0.1448	0.1361	0.1636	0.1566	0.1856	0.2709	**0.2959**	+8.4%
	H@10	0.0755	0.1998	0.2146	0.2415	0.2347	0.2646	0.2593	0.2656	0.2581	0.3029	0.4171	**0.4503**	+7.3%
Steam	N@5	0.0477	0.0744	0.0717	0.0945	0.1370	0.1613	0.1131	0.1727	0.3252	0.1842	0.2863	**0.3566**	+9.7%
	H@5	0.0805	0.1177	0.1203	0.1517	0.2171	0.2391	0.176	0.2559	0.4155	0.2710	0.3986	**0.4384**	+5.5%
	N@10	0.0665	0.1005	0.1026	0.1026	0.1283	0.1802	0.1484	0.2147	0.3557	0.2261	0.3332	**0.3875**	+8.9%
	H@10	0.1389	0.1993	0.2169	0.2551	0.3313	0.3594	0.2870	0.3783	0.5239	0.4013	0.5437	**0.5661**	+4.1%
ML-1m	N@5	0.0416	0.1903	0.1146	0.2885	0.3196	0.3705	0.3832	0.3980	0.4243	0.4454	0.4615	**0.5035**	+9.1%
	H@5	0.0715	0.2866	0.1932	0.4297	0.4673	0.5103	0.5353	0.5434	0.5755	0.5876	0.6025	**0.6351**	+5.4%
	N@10	0.0621	0.2365	0.1640	0.3439	0.3627	0.4064	0.4268	0.4368	0.4641	0.4818	0.5003	**0.5447**	+8.8%
	H@10	0.1358	0.4301	0.3477	0.5946	0.6207	0.6351	0.6692	0.6629	0.7008	0.6970	0.7219	**0.7422**	+2.8%
ML-20m	N@5	0.0511	0.1332	0.0771	0.2239	0.3090	0.3630	0.2538	0.4208	0.5134	0.4967	0.5058	**0.5666**	+10.3%
	H@5	0.0805	0.2128	0.1358	0.3601	0.4657	0.5118	0.3804	0.5727	0.6499	0.6323	0.6528	**0.6931**	+6.1%
	N@10	0.0695	0.1786	0.1271	0.2895	0.3637	0.4087	0.3062	0.4665	0.5440	0.5340	0.5398	**0.6189**	+13.7%
	H@10	0.1378	0.3538	0.2922	0.5201	0.5844	0.6524	0.5427	0.7136	0.7606	0.7473	0.7579	**0.8015**	+5.3%

Table 2: Ablation analysis in ML-1M and Beauty.

Variants	ML-1M		Beauty	
	HR@10	NDCG@10	HR@10	NDCG@10
w/o Contextual	0.731 (-1.4%)	0.536 (-1.5%)	0.422 (-6.2%)	0.276 (-6.4%)
w/o Regularization	0.730 (-1.6%)	0.537 (-1.3%)	0.425 (-5.5%)	0.279 (-5.4%)
w/o Self-Attention	0.621 (-16.3%)	0.483 (-11.2%)	0.352 (-21.7%)	0.183 (-37.9%)
w/o Max Pooling	0.611 (-17.6%)	0.503 (-7.5%)	0.339 (-24.6%)	0.166 (-43.7%)
Default	**0.742**	**0.544**	**0.450**	**0.295**

embeddings in our framework enable the model to capture fine-grained interest distributions in user behavior sequences and better interpret hidden relations among clicked items, which contributes to superior ranking accuracy of Seq2Bubbles than other models using point embedding.

5.3 Ablation Study

In order to investigate the effectiveness of some proposed units in our method and verify their necessity, we design a series of ablation studies for Seq2Bubbles. Here we consider four variants that simplify Seq2Bubbles in different ways: 1) removing the context-aware bubble embedding, 2) replacing the self-attention networks in the sequential model by average pooling, 3) replacing the max-pooling-based similarity measure as hard selection of bubbles (as in (11)), 4) removing the self-supervised regularization term. We report the results of HR@10 and NDCG@10 in ML-1M and Beauty in Table 2. The results show that the context-aware bubble embedding is relatively an auxiliary supplement which can bring up slight improvement for HR and NDCG. The self-attention network plays an important role in our model and the model performance would degrade significantly without it. This result conforms to the intuition that the self-attention unit in our framework can help to filter out noise in input sequence and capture temporal dependency which would be useful information for subsequent decoding layer. Furthermore, the max-pooling-based similarity computation is indispensable for a decent performance. When we replace it by hard selection as in (11), the HR/NDCG decrease by 23.5%/32.6%. Hence, our proposed approach for similarity computation turns out to effective.

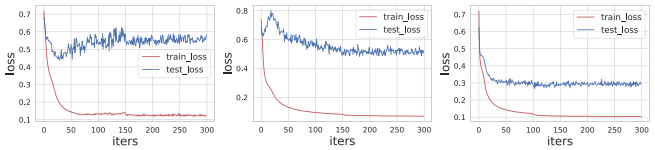
Finally, we study the efficacy of the self-supervised regularization approach. In Table 2 we can see that the regularization term helps to enhance test performance. As a further investigation, we present in Fig. 5 the learning curves of test loss with and without the regularization. In Fig. 5(a), we train the model without regularization. The test loss goes down at early stage and then goes up, keeping a large gap from training loss, which indicates over-fitting. In Fig. 5(b) and (c), we train the model using regularization with different t's. We can see that with proper setting $t = 1$ the gap between training loss and test loss become smaller, which indicates that the regularization can help to alleviate over-fitting.

5.4 Robustness Analysis

We further study the impact of corrupted input sequences for Seq2Bubbles in order to analyze its robustness w.r.t. noise in data. In Fig. 6, we compare Seq2Bubbles with its simplified version that replaces the bubble embedding by point embedding (for output) as used in previous models, and consider two different situations: (a) randomly removing several items in each sequence; (b) randomly replacing several items by other items in each sequence. As shown in the figures, when we increase the corrupted ratio, the performance of two models exhibits a drop. By contrast, NDCG of the baseline model drops much more significant than Seq2Bubbles, especially when the number of corrupted items are more than 20. The results demonstrate that the bubble embeddings can help to enhance model robustness w.r.t. noisy input sequences. As a matter of fact, point embeddings would presumably be more sensitive to random noise in sequences since the output state depends on all the items, while bubble embeddings would only change in partial regions w.r.t. variation of inputs.

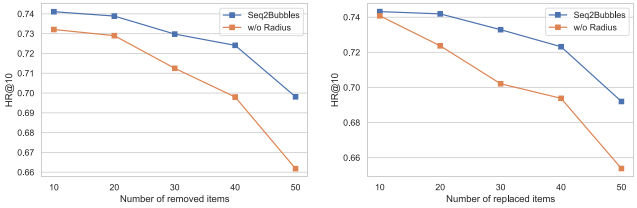
5.5 Parameter Sensitivity

We study the performance variation of our model w.r.t some hyper-parameters and the results are presented in Fig. 7. In Fig. 7(a) we show the HR@10 and NDCG@10 of our model with regularization



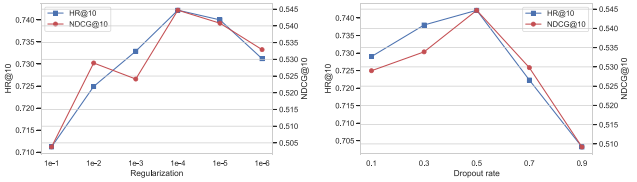
(a) w/o reg. (b) w/ reg. ($t = 10$). (c) w/ reg. ($t = 1$).

Figure 5: Learning curves of Seq2Bubbles with and without contrastive regularization.



(a) Removing items. (b) Replacing items.

Figure 6: Performance comparison of Seq2Bubbles w/ and w/o bubble radius w.r.t. synthetic noises in sequences.



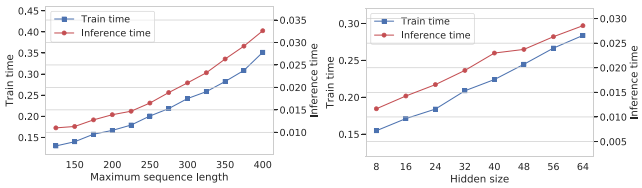
(a) Regularization weight λ. (b) Dropout rate.

Figure 7: Performance of Seq2Bubbles on ML-1M dataset when w.r.t different hyper-parameters.

weight λ ranging from 1e-6 to 1e-1. As we can see, the model performance first increases and then decreases, which again shows that a proper setting for λ can bring up satisfactory performance. If λ is too small, it cannot provide any regularization effect; if it is too large, the regularization would constrain model learning. Also, in Fig. 7(b), we change dropout probability from 0.1 to 0.9. The model performance also first goes up and goes down. The best HR and NDCG are achieved when we set it as 0.5. The dropout operation aims to reduce over-fitting by adding randomness in training. Too large dropout probability would lead to under-fitting while too small value would make little difference. While it has similar effects as our contrastive regularization, it cannot provide self-supervision by consistency. Overally, the performance variation of Seq2Bubbles w.r.t. λ and dropout probability is not much, which indicates our model is not sensitive to those hyper-parameters.

5.6 Scalability Test

We also discuss the scalability of Seq2Bubbles w.r.t. maximum sequence length and embedding dimension in Fig. 8. Here we statistic



(a) Maximum sequence length. (b) Hidden size.

Figure 8: Effects of maximum sequence length and hidden size on training and inference time of one step.

training time and inference time per mini-batch of data. The experiments are done with one NVIDIA Tesla V100 with 16 GB memory. As we can see, when we increase the embedding size from 8 to 64, the training time and inference time increase linearly. Besides, when we increase the maximum sequence length from 100 to 400 in ML-1M, the training time and inference time increase in an approximately linear trend. In fact, the feed-forward time complexity of Seq2Bubbles is $O(n^2d + nd^2)$ (where n is sequence length) given by self-attention and fully-connected networks. However, the computation of both is parallelizable with GPUs, which contributes to approximately linear scalability w.r.t. two variables in our case.

6 CONCLUSIONS

In this paper, we propose Seq2Bubbles, a new representation learning model for user behaviors in sequential recommendation. The model aims at embedding a sequence of clicked items into a set of closed region in vector space that characterizes user interests over potential items in the candidate sets. Our approach can effectively model complex distributions of user interests behind observed sequences and is shown to provide more satisfactory recommendation results by our experiments.

Potential Impacts. The key idea of our region-based embedding approach has broader applicality and potential impacts, instead of being limited in sequential recommendation studied in this paper. For example, real-world user behavioral data are often collected from multiple sources and composed of various modality, which induces complicated latent distributions. The bubble embeddings can be extended to such cases as a plug-in components to enhance representation power by replacing the traditional embedding module. Also, we believe that our approach can inspire a new paradigm for sequence data modeling in broad areas of knowledge discovery and data mining and further facilitate next generation of representation learning based on geometrically-inspired embeddings.

7 ACKNOWLEDGEMENT

This work was supported by the National Key R&D Program of China [2020YFB1707903], the National Natural Science Foundation of China [61872238,61972250], Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102), the Tencent Marketing Solution Rhino-Bird Focused Research Program [FR202001], and the CCF-Tencent Open Fund [RAGR20200105].

REFERENCES

[1] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. 2020. A Simple Framework for Contrastive Learning of Visual Representations. In *ICML*. 1597–1607.

[2] Xu Chen, Hongteng Xu, Yongfeng Zhang, Jiaxi Tang, Yixin Cao, Zheng Qin, and Hongyuan Zha. 2018. Sequential Recommendation with User Memory Networks. In *WSDM*.

[3] Katrin Erk. 2009. Representing words as regions in vector space. In *CoNLL*. 57–65.

[4] Shanshan Feng, Xutao Li, Yifeng Zeng, Gao Cong, Yeow Meng Chee, and Quan Yuan. 2015. Personalized Ranking Metric Embedding for Next New POI Recommendation. In *IJCAI*. 2069–2075.

[5] Ruining He, Wang-Cheng Kang, and Julian McAuley. 2018. Translation-based Recommendation: A Scalable Method for Modeling Sequential Behavior. In *IJCAI*. 5264–5268.

[6] Ruining He, Wang-Cheng Kang, and Julian McAuley. 2017. Translation-based Recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems (RecSys)*.

[7] Ruining He and Julian J. McAuley. 2016. Fusing Similarity Models with Markov Chains for Sparse Sequential Recommendation. In *ICDM*. 191–200.

[8] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *WWW*. 173–182.

[9] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent Neural Networks with Top-k Gains for Session-based Recommendations. In *CIKM*. 843–852.

[10] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based Recommendations with Recurrent Neural Networks. In *ICLR*.

[11] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets. In *ICDM*. 263–272.

[12] Jin Huang, Wayne Xin Zhao, Hongjian Dou, Ji-Rong Wen, and Edward Y. Chang. 2018. Improving Sequential Recommendation with Knowledge-Enhanced Memory Networks. In *SIGIR*. 505–514.

[13] Santosh Kabbur, Xia Ning, and George Karypis. 2013. FISM: factored item similarity models for top-N recommender systems. In *KDD*. 659–667.

[14] Wang-Cheng Kang and Julian McAuley. 2018. Self-Attentive Sequential Recommendation. In *ICDM*. 197–206.

[15] Yehuda Koren. 2008. Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model. In *KDD*.

[16] Yehuda Koren. 2009. Collaborative filtering with temporal dynamics. In *KDD*. 447–456.

[17] Yehuda Koren, Robert M. Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (2009), 30–37.

[18] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural Attentive Session-based Recommendation. In *CIKM*. 1419–1428.

[19] Jiacheng Li, Yujie Wang, and Julian J. McAuley. 2020. Time Interval Aware Self-Attention for Sequential Recommendation. In *WSDM*. 322–330.

[20] Hu Liu, Jing Lu, Xiwei Zhao, Sulong Xu, Hao Peng, Yutong Liu, Zehua Zhang, Jian Li, Junsheng Jin, Yongjun Bao, and Weipeng Yan. 2020. Kalman Filtering Attention for User Behavior Modeling in CTR Prediction. In *NeurIPS*.

[21] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: Short-Term Attention/Memory Priority Model for Session-based Recommendation. In *KDD*. 1831–1839.

[22] Jianxin Ma, Chang Zhou, Hongxia Yang, Peng Cui, Xin Wang, and Wenwu Zhu. 2020. Disentangled Self-Supervision in Sequential Recommenders. In *KDD*. 483–491.

[23] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *SIGIR*. 43–52.

[24] Yu Meng, Jiaxin Huang, Guangyuan Wang, Chao Zhang, Honglei Zhuang, Lance M. Kaplan, and Jiawei Han. 2019. Spherical Text Embedding. In *NeurIPS*. 8206–8215.

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

[26] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing Personalized Markov Chains for Next-basket Recommendation. In *WWW*.

[27] Ruslan Salakhutdinov and Andriy Mnih. 2007. Probabilistic Matrix Factorization. In *NeurIPS*. 1257–1264.

[28] Tiago Santos, Simon Walk, Roman Kern, Markus Strohmaier, and Denis Helic. 2019. Self- and Cross-Excitation in Stack Exchange Question & Answer Communities. In *WWW*. 1634–1645.

[29] Weiping Song, Zhiping Xiao, Yifan Wang, Laurent Charlin, Ming Zhang, and Jian Tang. 2019. Session-Based Social Recommendation via Dynamic Graph Attention Networks. In *WSDM*. 555–563.

[30] 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 *CIKM*. 1441–1450.

[31] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. In *ICLR*.

[32] Jiaxi Tang and Ke Wang. 2018. Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding. In *caser*. 565–573.

[33] Naftali Tishby and Noga Zaslavsky. 2015. Deep learning and the information bottleneck principle. In *Information Theory Workshop*. 1–5.

[34] Michael Tschannen, Josip Djolonga, Paul K. Rubenstein, Sylvain Gelly, and Mario Lucic. 2020. On Mutual Information Maximization for Representation Learning. In *ICLR*.

[35] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation Learning with Contrastive Predictive Coding. *CoRR* abs/1807.03748 (2018).

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NeurIPS*. 5998–6008.

[37] Luke Vilnis, Xiang Li, Shikhar Murty, and Andrew McCallum. 2018. Probabilistic Embedding of Knowledge Graphs with Box Lattice Measures. In *ACL*. 263–272.

[38] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2018. RippleNet: Propagating User Preferences on the Knowledge Graph for Recommender Systems. In *CIKM*. 417–426.

[39] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J. Smola, and How Jing. 2017. Recurrent Recommender Networks. In *WSDM*. 495–503.

[40] Le Wu, Peijie Sun, Yanjie Fu, Richang Hong, Xiting Wang, and Meng Wang. 2019. A Neural Influence Diffusion Model for Social Recommendation. In *SIGIR*. 235–244.

[41] Mang Ye and Jianbing Shen. 2020. Probabilistic Structural Latent Representation for Unsupervised Embedding. In *CVPR*. 5456–5465.

[42] Feng Yu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. A Dynamic Recurrent Model for Next Basket Recommendation. In *SIGIR*.

[43] Chang Zhou, Jinze Bai, Junshuai Song, Xiaofei Liu, Zhengchao Zhao, Xiusi Chen, and Jun Gao. 2018. ATRank: An Attention-Based User Behavior Modeling Framework for Recommendation. In *AAAI*. 4564–4571.

[44] Guorui Zhou, Xiaoqiang Zhu, Chengru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep Interest Network for Click-Through Rate Prediction. In *SIGKDD*. 1059–1068.