RNNs in Sequential Recommender Systems

在 ICLR-2018 公开的投稿论文中，发现了一篇用 RNN 做 Session-based Recommendation 的文章，在目前盲审阶段得到了 6、4、8 的分数，据我推测很可能是 ICLR-2016 中提出 Session-based Recommendation 的同一作者团队在之前模型上提出的改进。在这里，我更关注了该文的参考文献，了解 RNN 在推荐系统领域的应用历程。先是一篇短文，A Dynamic Recurrent Model for Next Basket Recommendation，发表于 SIGIR-16。之后是发表在 RecSys-2017 的文章，Sequential User-based Recurrent Neural Network Recommendations，利用 RNN 对用户建模进行推荐。

短文链接，长文链接

引言：A Dynamic Recurrent Model for Next Basket Recommendation

从用户和商品之间的交互信息中，我们可以提取出用户的喜好特征，传统模型得到的是一个全局性（general interests）的信息。如果按时间顺序考虑用户的交互过程，利用 Markov Chain 的假设，得到用户的一个局部特征。将全局和具有时间动态性的局部特征结合起来，就可以进行更精准的推荐。

模型和流程

简要介绍如下：

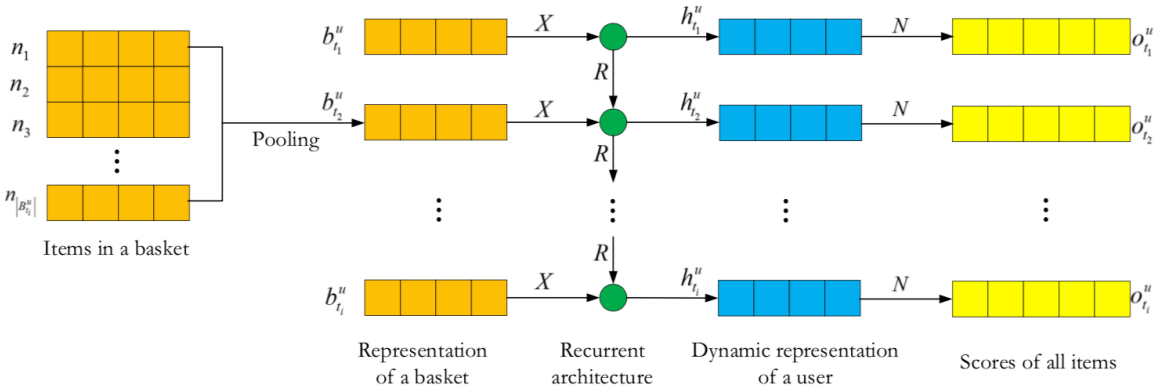

Figure 1: The framework of DREAM. Pooling operation on the items in a basket to get the representation of the basket. The input layer comprises a series of basket representations of a user. Dynamic representation of the user can be obtained in the hidden layer. Finally the output layer shows scores of this user towards all items.

1. 用户有若干次购物篮中的物品记录，每个物品都有一个 embedding 表示，对于每个购物篮的 embedding 是通过 max-pooling / average pooling 得到的；
2. 每个购物篮的表示，先通过矩阵 X 作用，作为 RNN 网络的输入后，计算得到具有用户特征的隐状态 h^u；
3. 隐状态 h^u 与物品矩阵 N （一个隐状态维数 * 物品数）得到该用户对每个物品的预测打分。

模型的优化目标

从经验上来看，采用的是 BPR-Loss，即保证正样本的得分比负样本的得分尽可能大的多。负采样的原则是负样本不是在当前 batch 中出现的物品，并且通常 0-1 隐式反馈选择这样的损失函数来优化有较好的效果。

上述模型中，每一个隐状态都可以预测下一个的候选 TOP-N 物品，从 t_{i-1} 时刻可以预测 t_i 时刻会加入到购物篮中的物品，很自然与时间相关，采用了 RNN 结构来建模。

小结

本文虽短，但是将 RNN 为何适用的 Intuition 说的比较清楚：旧方法中要么忽略了用户的兴趣随时间变化的情况；要么单单考虑了购物车中物品的变化，忽略了用户的全局特征，导致不同用户有相同的购物车物品会得出同样的推荐结果。RNN 的引入，最重要的价值就在于对用户特征在局部时间内的更为精准的学习。

基于用户序列的 RNN 推荐

摘要

RNN 在处理序列化信息上有很强的能力，这也使之可以用来生成序列化的推荐信息。推荐系统中，用户具有序列化的打分信息，每一个用户都可以对此生成一个 embedding 表示，用来预测下一个可能喜欢的物品。本文通过一种新设计的 Gated Recurrent Unit 来适应该任务，在离线实验中，比一般的 RNN 具有更好的效果。

简介

当前大多数的推荐系统或推荐算法，都认为用户的喜好在该系统内是固定的，包括协同过滤。然而，这种做法忽略了**用户消费商品时的时序相关性**。例如：有的商品基本只会在某个季节被购买；或者某些被购买的音乐、电影，都是和用户近期心情状态相关的；连载杂志等具有前后关系的物品也呈现先后购买的规律。相关的工作中，有一些把时间信息作为上下文特征直接融合到模型中（参考文献中有一个多维张量分解的推荐模型），或是借助其它非推荐相关的序列化模型。目前的多数研究，仍然忽略了推荐是一项时间相关的任务，还局限在提高预测的准确率上。

由于推荐系统问题的特性，序列化模型迟迟未被引入，很多机器学习的方法也都不一定适用。深度网络，让我们能够从数据中抽取特征，如一些栈式、层次化甚至是简单的表示学习方法。在自然语言处理、图像处理领域成功的方法，也很少在推荐系统中得到很好的扩展应用，仅有的一些也只是用在推荐系统的预处理中，提取一些辅助的特征。其他关注时序信息的做法，也有或多或少的局限性。

RNN 是深度学习中成功的序列化模型，为了在推荐系统中考虑时间这一重要因素，RNN 就十分典型了。最基本的 RNN 存在的问题是输入变量的比较有限（limited number of input variable），在此基础上提出的 LSTM、GRU 都通过门机制来控制信息流在网络中的传递，在实际应用中取得了更好的效果。但在推荐系统应用中，还没有针对性的门或其他方式来控制数据的传输，直接使用的话，往往没有什么效果，这是因为**没有考虑一个给出的购物序列是和一个特定的用户相关的**，他们的做法中，对于每个序列不加以用户的区分，很难捕获推荐任务中需要考虑的十分重要的用户特征。

本文针对上述问题，适应性地设计了一种新的 GRU ，对用户的消费行为进行序列化的动态建模；其次，门机制加上新的输入层设计，把用户的特性显式地融合在 RNN 中，可以对特定的用户进行个性化推荐。

相关工作

过去在推荐系统中的用户特征表示，认为用户的喜好在整个过程中是不变的，而实际生活中，用户的喜好则显示出较强的变化，因此，时间动态性在推荐系统中将被慢慢考虑进来。

1. **面向序列化推荐**

利用已有的一些显式、隐式反馈来抽取用户的特征是推荐系统研究领域中最基础的事情。而相关的研究表明，隐式行为通常比显式的打分数据更能反映出一个用户的行为习惯和喜好。用户的行为是动态的，使用一个静态模型去拟合特征，显然是不合理的，从而主要有两种考虑时间的做法：

- Time-aware，把时间作为一个上下文、补充信息融合到静态模型中，训练、预测时都有时间作为额外的输入，背后的原因是一个用户的行为有其个人的规律性，在多个时段内很可能具有同样的行为；
- Time-dependent，把用户的行为按时间顺序排序，时间信息在序列先后上体现，基于这个序列去做推荐，预测序列的变化

这些方法都是比较生硬的，Time-aware 方法像自然语言处理中一样，一些长距离依赖被忽略了；Time-dependent 则与本文的方法有共同之处，能够从序列中刻画一些诸如商品流行度变化的信息。

在处理时间信息上，大部分方法是对协同过滤模型的扩展，基于用户对当前物品的喜欢或不喜欢和他之前浏览过的商品有关的 Markov 假设。比如对很久之前购买的商品在训练过程中加一个惩罚项，减少它对模型中参数的更新的幅度（我：之前也这么干过，简单的指数衰减，有一定的效果）。其他的做法还有对用户消费记录中时间上比较接近的商品认为它们有一定的相似度，对商品的表示学习是一个很好的补充。

另一方面，矩阵分解在融合时间信息后，向高维扩展，可以做**张量分解**。总而言之，用到的思想就是把原本具有特征信息的、又与时间有关的物品特征，通过时间窗口或其他手段，放到一个序列中进行建模，同时考虑到了协同过滤和时间上的特性。

2. **深度学习在推荐系统中的应用**

   深度学习在推荐系统中刚刚开始显示出出众的能力。简单地，限制 Boltzmann 机，就可以对用户的打分进行预测。而更多地，是利用深度学习从外部数据中抽取补充特征，融合到基本的框架中去。很少有相关的工作把深度学习融合到与序列有关的推荐任务中。像上面的短文中提及的一样，过于关注序列，而忽略了这个序列是某个用户的序列，丢失了用户的特征。

   而要在序列中，融合用户特征，用深度学习的方法就几乎没有了。一种做法是，利用 RNN 从商品序列中学习出一个 embedding， 用户通过其他的信息也得到了一个 embedding，把这个用户 embedding 拼接到 RNN 生成的 embedding 上进行预测（这种做法用户信息只是作为一个补充，而没有对 RNN 的学习过程产生实质性的影响）。另一种做法，用户和商品根据各自的时序（商品的时序是什么？原文讲的不清楚）分别训练一个 RNN，两边的输出拼接在一起，再融合其他各种外部信息进行推荐和预测。这些做法用户特征、商品时序等的融合都比较松散，而且相对独立，没办法协同促进。

本文的模型

看起来独立的用户消费的商品，其实有一个序列化的信息蕴含其中，从中学习到的用户的喜好和行为的特征，能产生更高质量的推荐。作者提出的模型，通过精心设计的 GRU，把用户相关的信息“无缝”融合到 RNN 模型中。

1. **RNN 基本结构**

   商品特征按照时间顺序，经过 RNN 得到的隐状态，通过一个映射函数 T，把隐状态反映为对全局商品的一个打分，这与短文中的最后一个阶段是一致的：$\vec{o}^t = T\vec{h}^t$

2. **基于用户的 GRU 设计**

   为了将用户特征融合到 RNN 网络中来，我们的隐状态计算公式变为：$h^t = f(h^{t-1}, i^t, u^t; \theta)$，其中 i 代表商品的 one-hot 向量， u 是用户的 one-hot 向量。

- 线性用户特征融合

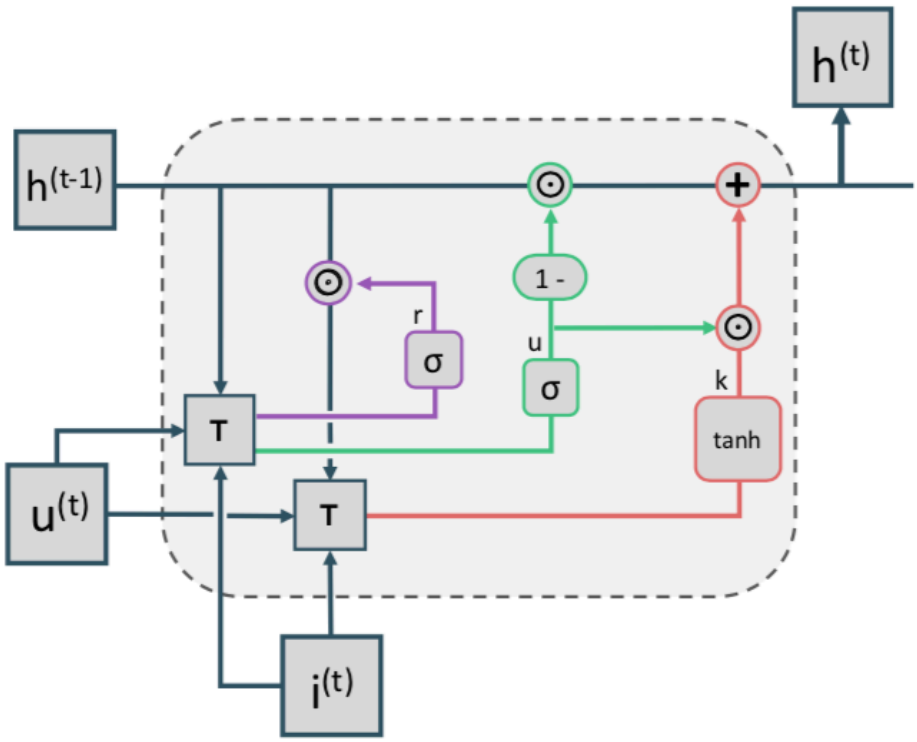

Figure 1: Linear user-based GRU cell: In addition to the item input vector i, the user representation v is included.

用户特征 u 可以视作输入层的补充，连接到 GRU 的门上，可以表示对当前商品输入的遗忘或者是更新。上图中绿色的 u 和紫色的 r 计算如下：

$$\begin{bmatrix} u \\ r \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \end{bmatrix} T_{3n,2n} \begin{bmatrix} h^{t-1} \\ E_i i^t \\ E_u u^t \end{bmatrix}$$

σ 是 sigmoid 函数，$T_{3n,2n}$ 表示从 $3n$ 维数到 $2n$ 维数的映射，E_i, E_u 分别是将商品和用户的 one-hot 向量映射到 n 维空间。用户的输入从而可以影响到 GRU 中的 update 和 reset 门。

红色的 k，是影响最终的隐状态的门，计算方式是
$k = \tanh(T_{n,n} r \odot h^{t-1} + T_{n,n} E_i i^t + T_{n,n} E_u u^t)$

最终的隐状态，$h^t = (1_n - u) \odot h^{t-1} + u \odot k$

  - 需要注意的是，这里的 i^t 是变化的，而 u^t 在模型中保持不变；
  - 为什么这么做，请参考 GRU 和 LSTM 原理，我也不是很懂，可以看看这个 slide。
- 修正（Rectified）线性用户融合，受到 ReLU 的启发

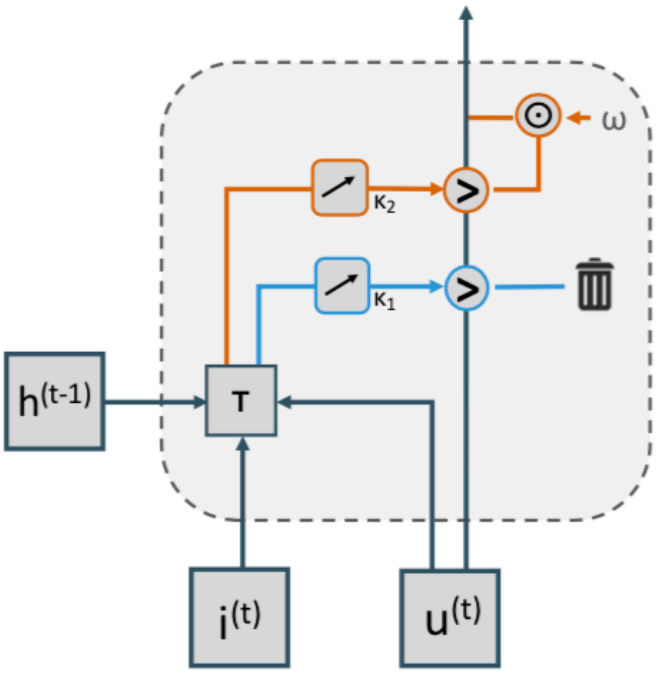

Figure 2: The rectified linear gating process detached from the complete cell: Item and user vectors as well as previous hidden state are concatenated and mapped linearly twice in order to calculate κ_1 and κ_2. Components of the user vector that are element-wise smaller than the corresponding values in κ_1 are discarded. Components smaller than κ_2 are diminished by element-wise multiplication with ω.

在上面的基础上，认为用户相关的参数在 RNN 训练的每一步都起到相同的作用。在上图中，由阈值 k_1 和 k_2 控制，对用户的向量 u 增加一个权重 $\omega \in [0,1]$ ，

$$\begin{bmatrix} \kappa_1 \\ \kappa_2 \end{bmatrix} = T_{3n,2n} \begin{bmatrix} h^{t-1} \\ E_i i^t \\ E_u u^t \end{bmatrix}$$

$$(E_u u^t)_l \leftarrow \begin{cases} 0, & \text{if } (E_u u^t)_l < \kappa_{1,l} \\ \omega (E_u u^t)_l, & \text{if } \kappa_{1,l} < (E_u u^t)_l < \kappa_{2,l}, \\ (E_u u^t)_l & \text{else.} \end{cases}$$

最终保证只有某些和用户表示比较相关的部分被输入进网络，对于用户的某些突发情况，可以避免带来过多的干扰。

- 用户注意力机制融合

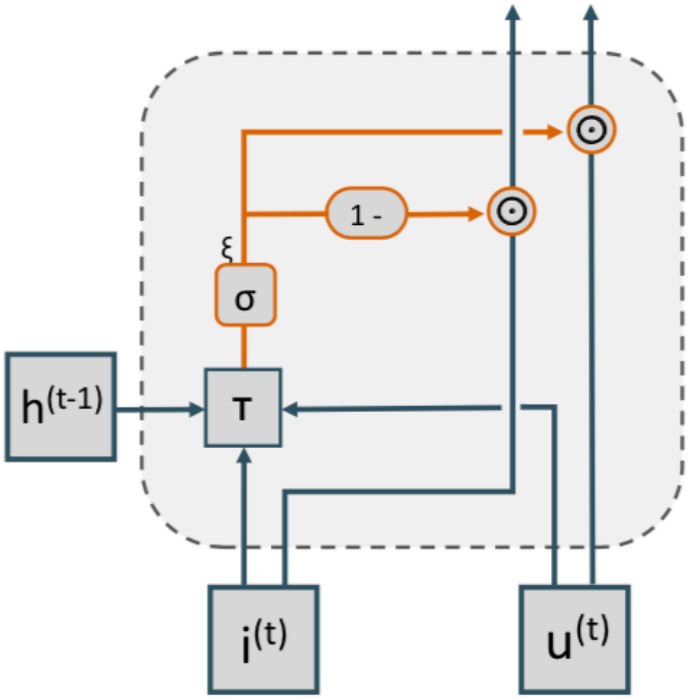

Figure 3: The attentional layer ξ detached from the complete cell: Item and user vectors as well as previous hidden state are concatenated and mapped with sigmoid squashing to form a gate that controls information flow between item and user component. Low sigmoids increase the throughput of item-related components, high sigmoids support the user side.

注：这里的图需要和第一张 GRU 图结合起来看！

如前面所述，用户的喜好被视作 RNN 网络中相对长期的特征，修正的线性融合方式，能够在某些物品上体现出长期中的一些小偏差问题上有一些控制作用。诚然，也可以用 Attention 来处理这种某些重要或是不重要的物品交互。在 GRU 中增加一个 **注意力正则门** （Attention Regulation Gate）：

$$\xi = \sigma(T_{n,n} h^{t-1} + T_{n,n} E_i i^t + T_{n,n} E_u u^t)$$

此时，计算

$$\begin{bmatrix} u \\ r \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \end{bmatrix} T_{3n,2n} \begin{bmatrix} h^{t-1} \\ (1_n - \xi) \odot E_i i^t \\ \xi \odot E_u u^t \end{bmatrix}$$

$$k = \tanh(T_{n,n} r \odot h^{t-1} + T_{n,n}(1_n - \xi) \odot E_i i^t + T_{n,n} \xi \odot E_u u^t)$$

与第一张 GRU 的设计相比，通过一个注意力的门 ξ，选择是完全忽略用户特征还是拷贝用户特征。换言之，如此设计的门决定了用户与商品之间的对应程度。（个人理解，用户购买的某个商品对他个性的体现有多明显，与 NLP 中文本情感分析类似，体现出一个词对于用户情感强度的程度。）

实验评价

本文主要对比了上述三种 GRU 模型和传统 RNN / GRU 模型以及相关的 the-state-of-art 方法在 MovieLens 10M 和 LastFM 数据集上的表现，评价指标是 MRR@k 和 Recall@k，这里 k 取 20。

Table 1: Hyperparameter values for all algorithms used in our experiments on the two datasets.

Parameter	Value
Item-based k-NN	
Nearest Neighbors	100
Exp. Dec. Item-based k-NN	
Decaying Constant	1
Matrix Factorization	
Latent Factors	20
Iterations	30
Learning Rate	0.01
Seq. Matrix Factorization	
Window Size	2

Parameter	Value
General GRU	
Batch size	1000
Layers	1
Hidden Units	1000
Unfold Dimension	20
Dropout	0.8
Gradient Cap	5.0
Iterations	10
Learning Rate	0.001
User-based GRU	
User Dropout	0.5

Table 2: MRR@20 and Recall@20 for all baselines as well as our proposed user-based RNN variants.

	MovieLens		LastFM	
	MRR@20	Recall@20	MRR@20	Recall@20
Baselines				
Item-based k-NN	0.036 39	0.121 42	0.044 38	0.101 26
Exp. Dec. Item-based k-NN	0.042 31	0.128 53	0.055 94	0.167 24
Matrix Factorization	0.011 92	0.077 44	0.032 89	0.127 23
Seq. Matrix Factorization	0.015 50	0.107 30	0.027 69	0.112 24
Standard GRU	0.047 30	0.157 73	0.143 74	0.202 69
User-based RNN				
Linear User-based GRU	0.052 51	0.160 28	0.182 52	0.249 20
Rectified Linear User-based GRU	0.059 01	0.186 97	**0.191 54**	0.254 09
Attentional User-based GRU	**0.062 55**	**0.205 40**	0.187 31	**0.254 85**

1. **参数设置**

   这里不太理解的是 unfold dimension 的意思，原文：with hidden dimensionality 1000 that is unfolded for 20 time steps. （猜测：RNN 序列的长度）其次，作者表示对于多层 RNN 在这里似乎并没有性能的提升，其他的都是一些经验设置。

2. **实验结果与分析**

   可见，在两个数据集上，通过精心设计的 GRU 结构的模型取得了更好的效果，主要在于融合进了用户的喜好特征，这也是能够比简单的 GRU 和其他方法提升更多的准确率的原因。已有相关实验能够验证，RNN 能够抓住一个序列中的时间特性及其效应，具有更好的预测模型。

   本文提出的方法，优于其他的方法，一方面是通过深度学习融合了用户信息，体现出个性化的要求；另一方面，巧妙的网络设计使得用户特征可以在时间序列上产生积极的影响。

   从 LastFM 数据集上有相对更高的提升可以看出，用户在听音乐时，切换的歌曲或者收听的音乐列表是体现出他当前的心情的，在此期间内，用户的心情是相对稳定的，而之后又会发生变化，体现出该模型动态对用户建模的优越性。此外，由于通常歌曲的时间比较短，所以在一定时间内“消费”的音乐数量比较多，这些音乐构成的序列体现出的特征比较一致，而电影则只有一两部，时间长了有一定的跳跃，在音乐的数据集上会具有更好的效果。然而，还是体现出用 RNN 建模能够体现出在一定时间内用户与商品之间交互的某种关联。

   对用户特征增加修正的融合、增加注意力机制都取得更好的结果，体现出用户特征在融合时，需要增加一定的数据的控制，使之尽可能带来增益，因此也不能太过。

结论与点评

本文在传统 RNN 在推荐系统的应用中，进行了有一定依据的修改，说明了 RNN 这样的网络模型可以从时序相关的数据中抽取更为准确的特征。把时间作为推荐系统应用的第一特征，对用户的行为建模，并且从用户的消费序列中抓住用户和商品之间的隐藏关联。

而 RNN 的各种变体，通常是任务导向的变体，能在实际任务中发挥更好的作用。其他诸如增加其他数据的输入，通过门的机制来控制这些额外数据在整个网络中的数据流向，从而使网络最后的输出能得到一个用户的个性化特征，即便用户 embedding 在底层是不变的，但是用门和注意力机制可以使这个 embedding 在序列的每个位置都起到不同的作用：被忽略或是被保留、部分保留，从而刻画出一种动态的用户特征融合方式。

作者提出的未来展望，包括把一些基于矩阵分解方法的改进引入到这个 GRU 网络中，但我觉得也就是写写而已，可能不会有什么后续了。我一直有的想法是，不对 RNN 本身的 Cell 做过多的改进，改变网络的输入结构（融合用户特征）、改变模型初始化参数或者是经典的 NLP 中的 Attention 机制等，都可以融合进来改进，这也是我目前改进的方向。此外，我觉得关于时间的信息，如果用户在两个商品上的

交互时间间隔很远了，就不应该作为序列的前后，所以我觉得另一个可以改进的地方是建立较好的用户交互序列，把某些明显的噪音去除（Attention 机制或许可以办到）、时间间隔远的可能就不适合了，究竟多久的间隔才适合作为一个序列数据或许也可以从数据集中学出来。总之，RNN 的火已经开始在推荐系统中烧起来了，任务导向对 RNN 做出一定的修改使之适合我们的任务，结合一个强烈的动机，有理论依据和实验验证，将成为这方面研究的套路。