Long and Short-Term Recommendations with RNNs

accepted by UMAP'17，原文链接

摘要

在 Session-based 推荐任务中，RNN 是目前最为有效的模型了，在于它对于协同过滤理论在序列化预测上的扩展。随着这种新手段的出现，我们可以对用户进行长期和短期的推荐，这在过去常被混在一起。本文首先将协同过滤方法在长短期推荐的任务上改造，同时引入 RNN 来更好地解决这一问题，证明了 RNN 模型在序列化推荐任务上的优越性。

简介

协同过滤算法认为用户和物品的特征都是静态的，是一种长期的特征，我们认为它能够刻画用户或商品的“一般”特征。即便是 timeSVD++，使用了时间的信息，也并没有很好地利用序列中先后出现的特征。现在，我们对问题重新建模，抽象成一个序列化预测的问题：给定一个用户访问过的商品序列，预测下一个会点击的商品，即一个典型 CTR 的问题。这么做的动机自然是数据集的特性，数据集中的每一条记录只是用户的最后一次或几次消费记录，一方面数据稀疏，另一方面只用这些最终的数据，丢失了访问记录的信息。序列信息的引入，得以弥补这些不足。

本文主要论述了以下两点：

- 在一些不那么稀疏的数据集上，RNN 模型表现强势，因为真的能够使用序列中的隐藏信息，学习用户喜好的变化过程，在某个时刻，用户对某种类型的商品不再感兴趣；
- 将序列推荐分为长期和短期的预测，短期推荐的往往是“下一个”会点击的物品，长期推荐则是指“最终”会购买的物品。这在静态推荐算法中并不加以区分，因为并不存在序的情况，而在动态的序列化推荐中，短期的推荐算法的设计中，都是尽可能准确预测下一个物品，而代价可能是对于长期的物品预测不够准确。

> 【例】
>
> 短期推荐：根据最近的播放列表预测用户 **下一首** 会播放的歌，即序列 12345 预测出下一个是 6；
>
> 长期推荐：根据用户读过的 **所有** 的书来推荐一本其他的书，这本书可能不会马上购买，会在买了其他的书后买，即序列 123456 能预测出 9 将会出现。

本文的主要贡献总结如下：

- 对长期和短期数据进行可视化划分，并在不同的推荐系统算法中进行了多角度的对比；（注：文中说的不是很多，主要体现在实验设置和实验结果中）
- 对 RNN 模型进行了改进，对长期和短期物品推荐的性能 trade-off；
- 对短期推荐的预测多样性进行了探索。（注：在实际推荐系统中，候选推荐的多样性也是评价的一个指标）

本文的相关代码：Github 地址，数据集（train/valid/test 共1.2G）也可供下载。

相关工作

推荐系统任务也可用一个序列化预测问题来建模，如 Markov Chain 思想的应用。早期就有一些推荐系统的专家和大佬使用 Markov 模型、序列化模式挖掘（ Sequential pattern mining) 等，做出了比最近邻推荐算法更好的结果。MDP 理论也开始逐渐应用，首次把这种思想应用到推荐系统的相关工作中时，虽然不是直接生成候选推荐物品（这一相关工作的重点是在此基础上使用一些启发式方法来提高聚类的性能）。

之后这项工作失去了关注，只有 FPMC (Factorizing Personalized Markov Chain for Next-basket Recommendation)，在矩阵分解的基础上融合了 Markov 过程。还有 Fossil 方法，与 FPMC 思想类似，不过分解的不是用户-物品矩阵，而是物品之间的相似度矩阵。

直到最近，以 Hidasi 为开端，将 RNN 应用到这项任务中来，设计了两种损失函数（BPR 和 TOP1）来优化模型，此后还有一些对这个模型的优化。值得一提的是，Spotify 在线音乐平台上，开始使用序列化的推荐模型来为用户生成播放列表，但是由于歌曲数量过于庞大，基于 RNN 的模型难以处理，他们用的是一种多层 softmax 结构作为预测输出。

关于如何表示物品，既然有了消费的物品序列，也可以用 word2vec 学习一个更好的表示。有了这些表示，那么一个 session 就可以用 RNN 处理为一个向量，直接预测下一个序列中最可能的下一个物品。

CF + RNN

本文是近期最早将 RNN 与传统协同过滤方法结合起来构建的模型。

数据集

数据集的统计信息如下：

Dataset	Num of sequences	Items	Interactions
Movielens	6040	3706	1,100,209
Netflix	480,189	17,770	100,480,507
RSC15	7,981,581	37,486	31,708,505

这里重点介绍一下 RecSys'15 Challenge 的数据集，这里的每一条记录并不代表一个具体的用户，而是一个 Sequence，这里的 Sequence 长度中位数仅为 3，而 Movielens 达到了 96；此外其他两个数据集的时间戳上有很多是十分接近的，人为划分的 Sequence 可能不是十分合理，数据集的质量并不高。（注：我在相关研究中也有这种感受）从实际情况上来看，Sequence 的数量越多，每个 Sequence 越短，矩阵分解之类的方法就越没有信服力，因为对每个 Sequence 来说，如果有一个独立的表示才更合理，显然静态方法是做不到的。

RNN 设计

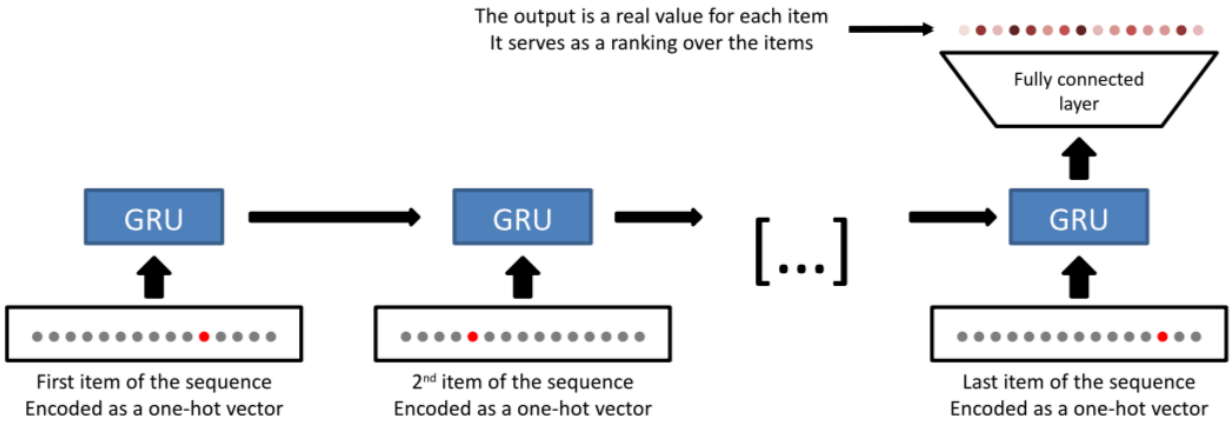


Figure 1: Schematic representation of the RNN.

基本原理不细说了，主要介绍用到的两种优化目标函数：CCE 和 Hinge。实验的参数设置上，基本上都是经验设置。

CCE，Categorical Cross-Entropy，即使用交叉熵损失，通过 log 和 softmax 操作计算。存在的问题是，softmax 与物品数呈线性关系，在大规模数据时性能不高。

Hinge Loss，与 SVD 的优化目标相似，与 CCE 相比，每个物品作为候选推荐的概率是独立的，没有做 softmax 操作，损失分为正确的候选（这个 Sequence 中后面所有的物品）和采样的“坏”的推荐候选（理论上包含 Sequence 中未包含的所有物品），对于正样本和负样本的差距尽可能大。（文中给出的公式似乎有问题？我觉得中间应该是加，求解释。）它的优势在于可以控制正样本的数量，作者的意思是说，正样本不一定只是 Sequence 中确定的下一个，也可以是下面的几个。

对比的方法

具体如何计算的不详细介绍了，不是重点，感兴趣的请阅读相关文献。

1. User-KNN，通过用户的 k 个最近邻来估计用户对其他物品的打分，从打分的高低预测是否会出现在 sequence 中；
2. BPR-MF，使用矩阵分解的方法，优化目标是 BPR 来计算用户对物品的打分；
3. MC 方法，对用户使用 Markov Chain 建模，转移概率直接从观测到的数据中进行统计，请自行类比 NLP 中的 Bi-gram 语言模型；
4. FPMC，分解用户-物品矩阵并融合 MC，上升到了张量的分解，没有仔细研究，是 2010 年的一项工作；
5. Fossil，与 FPMC 类似，同时受到 FISM 的启发，分解的是物品-物品的相似度矩阵。（在我看来，用户的固定特征似乎被丢掉了，或者说，这个 Sequence 中的所有物品，就用来代表了用户特征，与 RSC15 数据集的设定是一样的。）

模型评价指标

1. Recall，主要体现出模型在长期推荐预测上的准确率；
2. sps，Short-term Prediction Success，是对 Sequence 中下一个物品的预测成功率，成功即为 1，否则为 0；
3. User coverage，至少预测对 1 个物品的用户比例；
4. Item coverage，至少被正确推荐 1 次的物品比例，coverage 衡量推荐系统的全面性，是适应实际任务的指标；
5. Blockbuster Share，被正确推荐的物品数占 TOP 1% 物品的比例，衡量对长尾物品的推荐性能。

实验结果

Table 3: Comparison of top-N recommendation methods on Movielens 1M, Netflix and RSC15

		Metrics (@10)				
	Method	sps (%)	Item coverage	User coverage (%)	rec (%)	Blockbusters share (%)
Movielens	POP	5.0	50	65.6	3.62	97.73
	BPR-MF	12.44 ± 1.26	388.50 ± 14.32	82.94 ± 1.27	5.58 ± 0.12	44.08 ± 1.43
	User KNN	14.40	277	80.8	6.31	50.36
	MC	29.20	518	77.0	4.90	16.47
	FPMC	28.10 ± 0.71	614.00 ± 10.32	82.94 ± 1.45	5.70 ± 0.13	14.26 ± 0.71
	Fossil	22.46 ± 1.67	556.40 ± 39.81	83.04 ± 0.61	6.32 ± 0.14	11.22 ± 1.98
	RNN-CCE	33.45 ± 1.17	669.75 ± 15.58	87.73 ± 0.98	7.52 ± 0.14	13.91 ± 0.28
	RNN-Hinge	30.91 ± 2.15	611.56 ± 45.52	88.40 ± 0.82	7.72 ± 0.17	16.40 ± 1.92
Netflix	POP	5.92	54	68.2	4.65	100
	BPR-MF	9.93 ± 0.70	591.60 ± 16.36	79.19 ± 0.71	6.88 ± 0.09	73.84 ± 1.32
	User KNN	13.04	383	80.84	8.49	79.86
	MC	32.50	594	64.50	3.17	53.02
	FPMC	26.38 ± 0.80	542.56 ± 6.87	70.98 ± 0.46	3.75 ± 0.08	64.53 ± 0.51
	Fossil	24.26 ± 0.64	852.78 ± 28.32	76.93 ± 0.77	5.17 ± 0.12	40.07 ± 1.78
	RNN-CCE	41.00 ± 0.99	845.88 ± 26.00	82.43 ± 0.48	6.37 ± 0.16	53.15 ± 1.39
	RNN-Hinge	32.28 ± 0.87	935.20 ± 38.23	78.92 ± 0.65	5.76 ± 0.24	39.29 ± 0.43
RSC15	POP	1.24	11	2.24	1.2	100
	MC	39.50	2945	46.55	32.76	33.33
	RNN-CCE	52.78 ± 0.08	3536.67 ± 4.99	59.34 ± 0.10	44.52 ± 0.06	33.14 ± 0.17
	RNN-Hinge	35.12 ± 0.27	2374.33 ± 21.93	43.18 ± 0.16	30.24 ± 0.23	33.76 ± 0.30

- 首先，使用序列化信息的模型，要比不用的好很多；
- 还可以发现，具有高 sps 的模型，同时也具有很高的 coverage，在后面会具体分析；
- 在短期（即时）的推荐任务上，RNN 表现出更好的性能，由 sps 体现。

长短期推荐分析

从上面的实验结果中，发现短期的性能 （sps）得到了大幅提升，而对于较长期的物品推荐，Recall 值提升的并不明显，甚至本文的模型在 Recall 的性能上还不如具有低 sps 的 User-KNN 方法，这主要是由于长期推荐的任务中，缺少用户的固有特征。事实上，在一些音乐、视频网站中，我们需要短期、快速的推荐，更关注的是用户频繁变化的当前喜好，而在读书、门户信息推荐中，则通常只注重最后一次成功推荐的结果，要考虑的是用户长久以来表现出的喜好。这就有了长期、短期推荐之分。长期的推荐更注重对用户长久以来的所有记录中体现出的静态特征和长期偏好，而短期的推荐比较关注物品之间的变化的联系。

推荐系统的推荐预测本质上是一个排序，排序靠前的就推荐。那么如果输入序列够长的话，我们每次生成的候选推荐物品数 N 如果为1，就是典型的短期推荐（对应 sps 指标）当 N 增大时，就慢慢转变为一个长期推荐的情景。假设有一个用户有长为 2q 的序列，使用前 q 个物品作为输入，预测出 N 个候选物品，对这 N 个物品计算平均排序，av-r@N （注意，这里的 N 是根据实际情况的 q 决定的，对不同的序列可以取遍 1 到 q，并不是固定的，在进行长期推荐预测的话，取 q 即可），表示长期推荐预测的准确率。

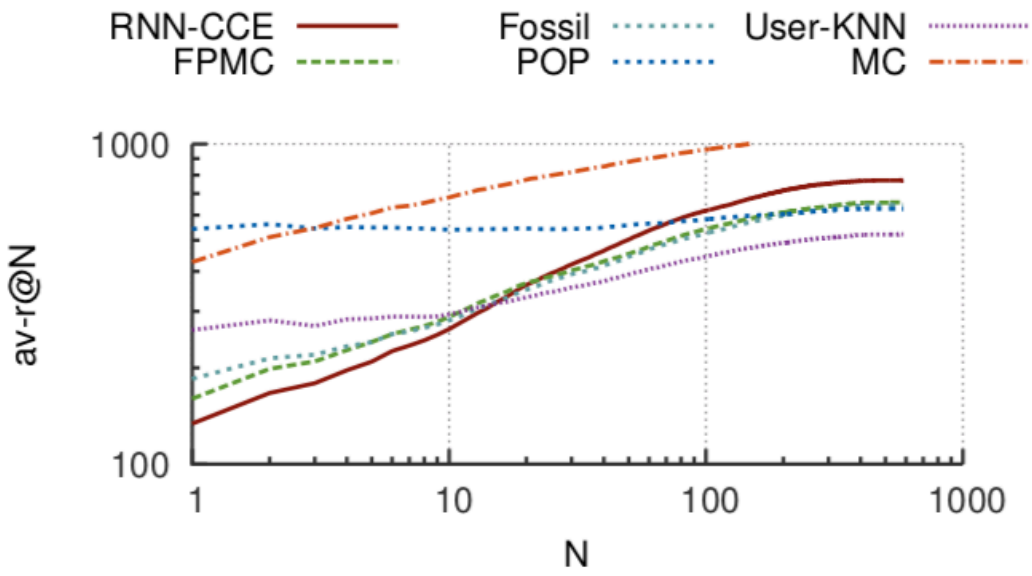


Figure 2: Short-term/long-term profile of RNN-CCE, User KNN, FPMC, Fossil, MC and POP on Movielens.

据此，作者做了一些对比试验来分析。总的来说，短期推荐的性能要比长期的好，POP 模型并不在意用户特征，所以表现没什么变化。但是 User-KNN 在长期推荐的性能上最好，甚至能够预测出序列 20 个物品之后的物品。为什么使用 RNN 的方法在长期推荐性能上较差？作者直接通过 Trade-off 把结果调上去，我认为还是 RNN 的特性，在很久之后才出现在序列中的物品，和前面出现的物品可能关系不大，在序列后面出现，更多的因素是因为用户的长期、静态、固有的特征导致的。

提升长期推荐的性能

RNN 模型在短期推荐上优异的性能，如何充分利用呢？为了使它准确预测不仅仅是“下一个”物品，而是“下几个”物品，作者提出了三种 Trade-off，两种是对训练过程进行的改进，还有一种则是对优化目标函数进行改进。

1. Dropout

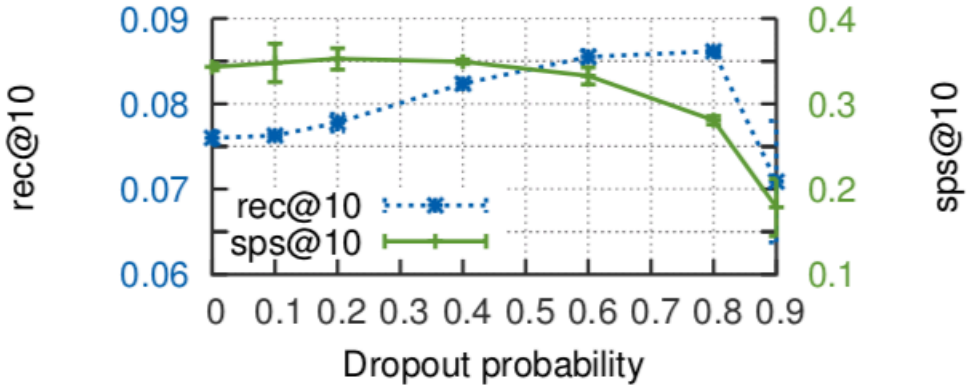
   对训练 Sequence 中的物品按概率随机的 skip，或者理解为丢弃，使我们能预测下面的几个物品。

2. Shuffle Sequence

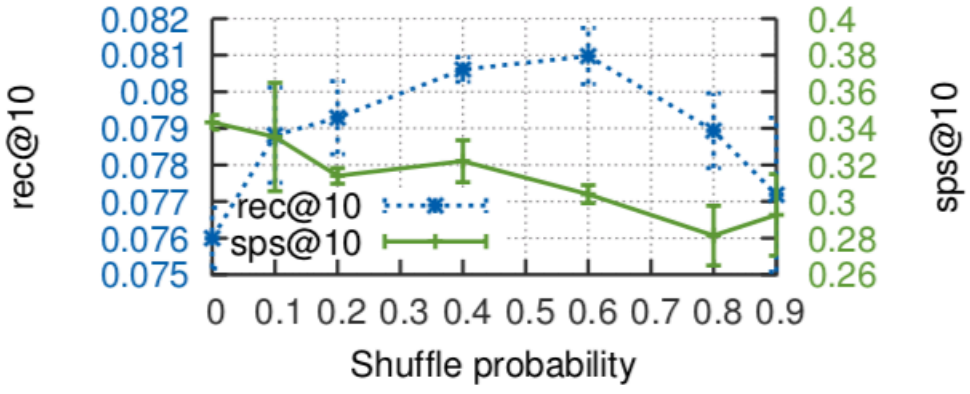
   对训练 Sequence 中的物品进行重排或修改，人为加入一点噪音，可以使模型抓住序列中的比较“大”的 **稳定** 静态特征。

3. Multiple Targets

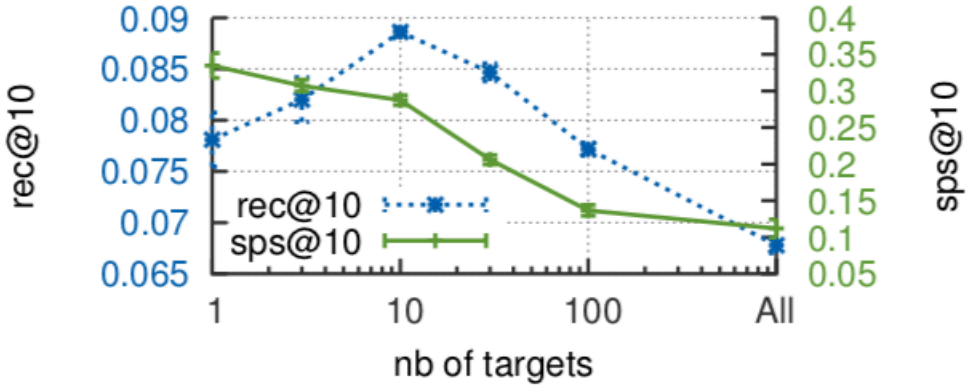
   原本的优化函数是针对短期推荐的性能设计的，尤其是 CCE Loss，只考虑序列的下一个计算损失。Hinge Loss 则可以包含更多的正样本，这正是他的优势，能够把序列中接下去的多个（实验表明 10 个比较合适）作为正样本。



(a) Dropout



(b) Shuffling



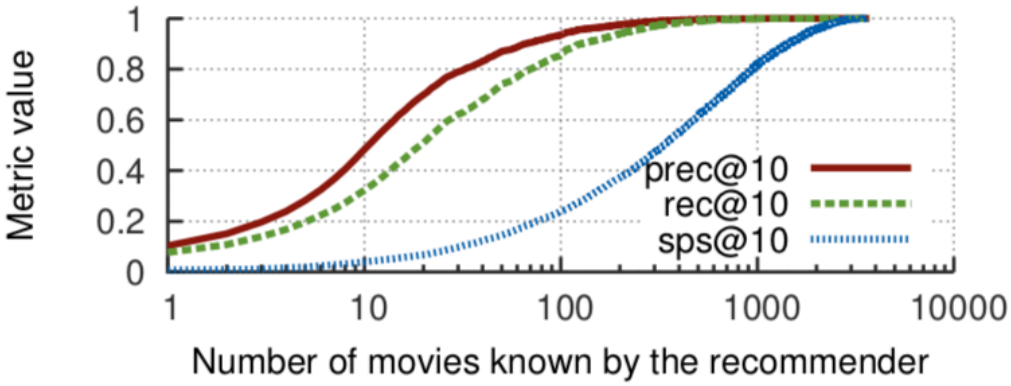
(c) Multi-targets, hinge loss

Figure 3: Influence of dropout, shuffle and multiple targets on short and long-term predictions on Movielens.

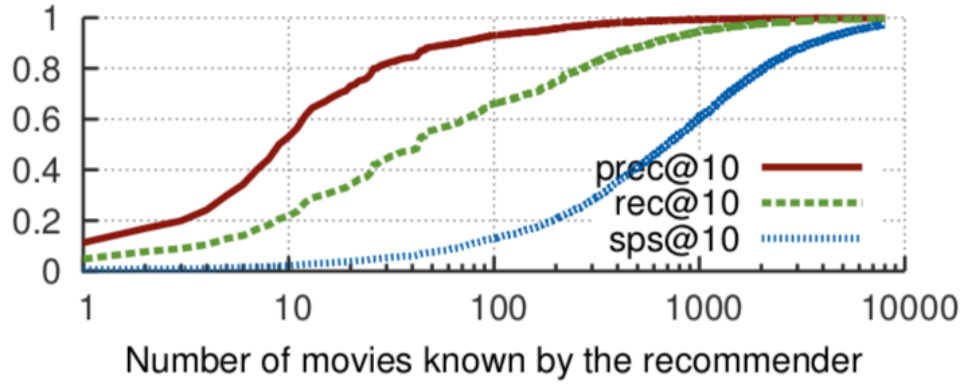
三种 Trade-off 在 Netflix 数据集上都提升了长期推荐的性能，体现在 Recall 指标的提升上，但仍低于 KNN 模型。总体来说，综合多个指标来评价，KNN 在推荐多样性上就明显逊色，RNN 模型还算是最优。

多样性推荐的再讨论

从 sps 的评价指标上来看，越高代表着推荐越多样化。User-KNN 等基于 POP 的方法更倾向于生成推荐流行物品，所以 Recall 值会比较高。换言之，优化短期推荐的目标，以推荐的多样性要求为代价。原因在于：sps 指标下正确的候选推荐物品是 Recall 指标下的子集，从而这里面的每一个物品对更多的用户来说都是正确的长期推荐结果。想象一下，如果为了优化 sps 结果（短期推荐正确），需要减少候选物品的个数，此时对于长期推荐来说，排序属于后半段的正确物品会被丢弃，从而 Recall 下降。反之亦然。



(a) Movielens 1M



(b) Netflix

Figure 5: Evolution of the rec@10, prec@10 and sps@10 for a recommendation that can only recommend the top t most popular items. The maximum value of the rec@10 is actually much smaller than 1 but we normalized the recall here by dividing it by its maximum value to make the graph more readable.

作者给出了一个简单实验进行验证：假设有一个 oracle 推荐系统，把它已知的所有物品都放到候选推荐中，当它已知的物品数量增加（我认为是根据流行度降序排列的顺序放入）时，precision 和 recall 都增加的很快，较早收敛（越来越多的流行物品被放入候选中，达到饱和，因为被消费的物品必定具有一定的流行度），而 sps 的增加十分缓慢，在后期的增长稍快，此时候选的物品很多，可以反映出多样性的特点，sps 指标刻画出推荐系统多样性的要求。

结论与点评

本文将 RNN 融合到序列化推荐工作中，先是设计了不同的优化目标。接着从现实应用中，分出长期和短期推荐的不同，按照多种推荐系统的性能评价进行多种 Trade-off，从实验数据和符合直观感受上解释了原因，使得本文的学术研究部分比较薄弱，偏工程性一些。

长期、短期推荐的不同，主要在于对用户、商品在两种情境下的不同建模，长期推荐更注重静态的特征，通常是一些不变或变化很小的特征，在建模时一般不会再改变，直接作为特征输入。短期的推荐则与用户交互的过程十分相关，更注重用户先前消费了什么，因此 RNN 被引入来处理、建模生成动态特征。我认为，一种简单的做法，在生成候选推荐的打分时，把长短期特征拼在一起，在长短期推荐的不同任务上，用具有不同参数的两套 attention、dropout 等方式处理看起来更为简洁：长期特征通过 attention 抓住已知序列中的共性和不变性，短期特征则由 RNN 负责刻画，如果从 case-study 中能证明有效的话。。。

【脑洞】长短期推荐的不同，是不是可以认为我在生成候选推荐的时候，还可以生成一个时间戳，对某物品计算候选推荐得分时，同时将相对时间间隔的时间戳放上去，如果该物品在较远的相对时间戳的得分高，就认为应该属于长期推荐的，如果时间较近的情况下得分高，就认为是短期推荐的物品。难点：如何设计这个相对时间戳?