Leverage Long and Short-term Information in Content-aware Movie Recommendation

由相关微信公众号推荐，也有一个知乎专栏，arxiv 链接。

本文在传统利用 RNN 解决 session-based recommendation 问题上，引入了一些用户和物品的静态信息，此为看点一。运用对抗网络训练模型，融合强化学习的思想，此为看点二。看点三略显平常，融合一些辅助信息缓解冷启动问题。

摘要

电影推荐系统，根据用户的口味生成一个候选推荐列表。目前主要有两种建模的立足点：长期静态模型和短期动态模型。前者从用户与电影的交互信息抽取出近似认为不会变化的用户和电影的特征，后者则是将用户兴趣和电影的信息在短期内进行一系列的调整与改变。本文提出了 LSIC 模型，使用对抗训练的方式，从数据中提取长、短期的特征：由生成器，结合强化学习的相关做法，根据用户的消费序列生成下一个可能的候选电影；而判别器负责区分生成器给出的结果和真实结果。此外，关于电影的一些外部信息也被引入，可以缓解冷启动问题。

简介

在大数据背景下，推荐系统的构建已经从人工设置特征转向数据驱动的，机器学习式的从数据中学习特征和规律的一项工作。推荐系统的主要任务是，从海量的潜在商品中，挑出最符合 **当前** 用户口味的，可谓 N 里挑一了。目前通用算法框架主要有：协同过滤、基于内容的过滤和这两者的综合。

矩阵分解因其简单、效果理想、易于扩展的优点，成为最成功的协同过滤模型之一，在于它可以从大量的交互数据中学习出用户和物品的潜在特征，一种蕴含在长期过程中的静态特征。但是，这种方法没有考虑到数据、用户、物品随时可变的自然性质。尤其是电影这样的物品，很容易受到外界因素的影响；用户也会产生喜好的微妙变化。对于这样的情况，我们可以采用 RNN 之类的框架，处理电影和用户的动态时间特征。RNN 模型，将推荐问题转化为一个序列化预测的问题——将最新观察到的用户交互记录作为输入，更新 Cell 中的状态，计算出新的状态特征进行预测。这种做法主要抓住了短期信息，相关的实验证明，可以提高推荐结果的多样性。

比较近期的工作中指出，可以融合矩阵分解和 RNN 模型，使它们在长、短期特征信息上的成功互相补充。此外，只利用用户和电影之间的历史交互信息，是十分稀疏的，通常我们拥有更为丰富的电影信息，如相关海报、介绍等，可以帮助推荐系统更好地理解用户和电影。从实际的角度来说，电影能否吸引一个用户，更多的是视觉冲击，用户更喜欢哪种风格相近、视觉效果类似的相关电影，而同一个演员参演的联系则会更加弱一些。这些后验信息的加入，可以进一步提升推荐系统的性能。

至此，本文标题中的 Long and Short-term Information 可以解释为利用矩阵分解、RNN 模型从数据中提取长、短期的特征，Context-aware 则是指利用了电影海报之类的辅助信息，构建一个 LSIC 推荐系统模型。该模型通过对抗网络，将矩阵分解和 RNN 序列化推荐的任务结合起来，目标是在一个 Siamese 网络中，判别器 D 无法分别由生成器 G 生成的候选推荐和用户实际选择的电影。

本文的主要贡献有以下几点：

- 第一次使用 GAN 各取 MF、RNN 模型的长处，并且能够自适应地平衡和调整长短期特征的作用
- 提出多种 soft / hard 融合的方式，对各种变体进行了对比，并在这个过程中，利用长期特征来引

导短期特征的学习
- LSIC 模型采用强化学习的思想，对于生成器 G 来说，可以利用 policy gradient 来更新不可导的任务评价指标函数
- 通过收集到的电影海报图片，从中提取出电影的 content 特征作为辅助，增强推荐系统的能力
- 在实际数据集（Netflix 和 Movielens）上测评，取得了 state-of-the-art 的结果

相关工作

与本文运用的相关技术，主要有三：矩阵分解、RNN 与序列化推荐、GAN 与推荐模型。

矩阵分解

从用户-电影的历史交互记录中，提取长期、稳定的特征，用户与电影的特征向量通过内积运算得到预测得分。该算法模型简单易实现，且有许多变体和改进。（我个人觉得这部分有的属于作者凑参考文献的）值得一提的是，除了关注已有的用户-电影的显示记录，隐式反馈也可以考虑进来，一方面可以防止模型过拟合，另一方面，直觉上可以刻画出用户不喜欢的一面。此外，矩阵分解方法也十分便于扩展，可以把电影的海报等图片信息融合进特征向量中。

RNN 与序列化处理推荐问题

矩阵分解出于用户和电影的特征基本不变的假设，这显然是不符合实际的，从而在 SVD 分解的基础上融合人工构造的时间特征，成为 TimeSVD++。最新的做法是使用 RNN 来处理序列化推荐的问题，以 Hidasi 等人的工作为代表，把用户在序列中的第一个交互动作作为初始状态，之后输入的每一个点击行为都会有一个输出状态，根据这个输出来生成一个候选推荐。Wu 等人运用 RNN 模型利用带有时间戳的信息，生成异构时间的推荐反馈【1】。另外一个 Wu 的 RRN 相关工作中，对用户和电影使用 LSTM 自回归模型，并结合了矩阵分解（MF）静态模块视角下的固定属性特征【2】。**与此不同的是，本文使用了对抗网络的方式融合了 MF 和 RNN 两个模块，在生成器部分，旨在生成有根据的、高质量的候选推荐列表。**为了缓解冷启动的问题，本文追随了 Cui 等人的 VT-RNN 工作，融合了电影的海报作为 Visual 特征，体现了 Content-based 推荐算法的贡献。

对抗网络与推荐系统

GAN 在图像生成、图片 caption 、序列生成等任务中取得了一定的效果，在 IR-GAN 中，通过迭代，优化生成检索和判别检索模块。IR-GAN 在搜索、物品推荐和 QA 等领域有较好的结果。本文与之的不同之处在于，GAN 框架下，还融合了 MF 得到的静态特征和 RNN 的动态特征处理机制。第二，IR-GAN 无法预测未来的行为，而推荐系统任务视角下，更关注预测用户未来的选择，甚至在 IR-GAN 中，存在使用未来的信息来估计过去行为的情景，这也是目前推荐系统强调的时效性和因果性。第三，本文还融合了外部特征，对 GAN 训练的框架进行了更为丰富的填充，借助 GAN 来提升推荐系统的性能。

本文模型

本文的问题描述如下：一个稀疏的用户-电影打分矩阵 R，其中用户数和电影数分别是 U 和 M，打分矩阵中的每一项 $r_{ij,t}$ 表示用户 i 对电影 j 在 t 时刻的打分，本文和其他的相关工作也大多数都使用 TOP-n 推荐任务，即不预测打分的偏差，而侧重用户对电影的选择。本文提出的 LSIC 模型在 GAN 框架下，采用强化学习的理论融合了 MF 和 RNN 两种不同的特征处理方式，同时应用用户、电影的长短期特征。模型图如下，下面分别介绍其中比较重要的模块。

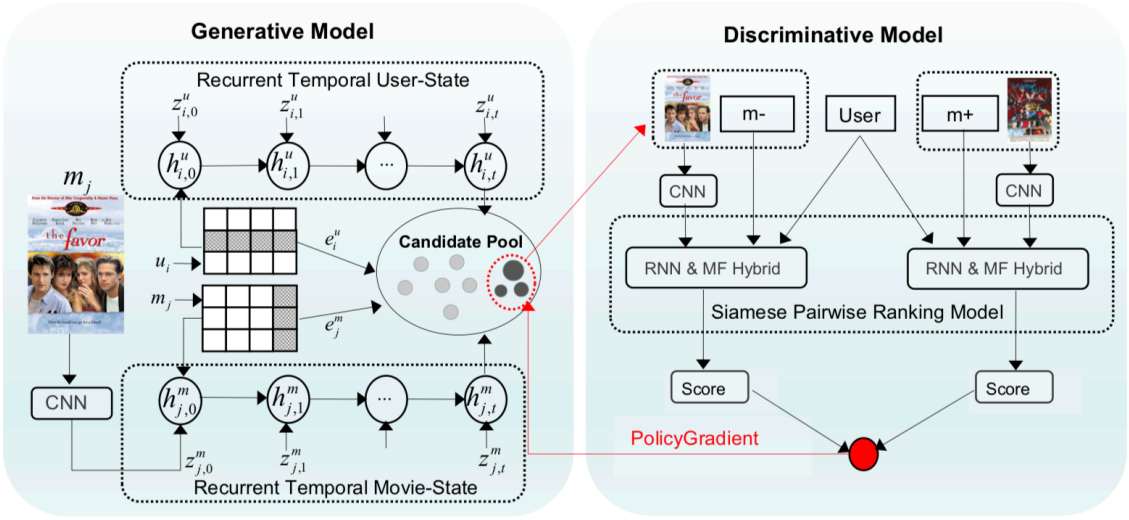


Figure 1: The Architecture of Long-term and Session-based Ranking Model with Adversarial Network.

矩阵分解，MF 模块

MF 框架刻画的是长期状态，或者说是用户和电影的全局信息特征 e^u 和 e^m。根据观察到的数据，对于用户 i 和电影 j 对，最小化损失：

$$\arg\min_{e^u,e^m} \sum_{i,j} I_{ij} \left(r_{ij} - \rho\left((\vec{e}_i^u)^T \vec{e}_j^m\right)\right)^2 + \lambda^u \|e^u\|_F^2 + \lambda_u \|e^m\|_F^2$$

这里的 I_{ij} 不是指示函数，当 $r_{ij} > 0$ 时，$I_{ij} = 1$，否则取 0，意在仅使用显式反馈。在这样的优化算法下，得到的是一个预测的打分，可以根据打分排序，给出一个推荐序列，可是效果不尽如人意，因为最小化的目标是均方误差，与 top-N 推荐问题下的排序损失不完全一致。

循环神经网络，RNN 模块

RNN 在推荐系统领域中，主要处理序列化信息，即一个时间段内的用户-电影交互记录，从中提取该时间段内的特征。这样处理的优势在于，可以根据已经发生的历史记录，动态调整生成预测的状态，更能体现用户在消费的动态特性。本文采用基本的 LSTM 设计，从用户和电影角度，设计两套平行的序列：

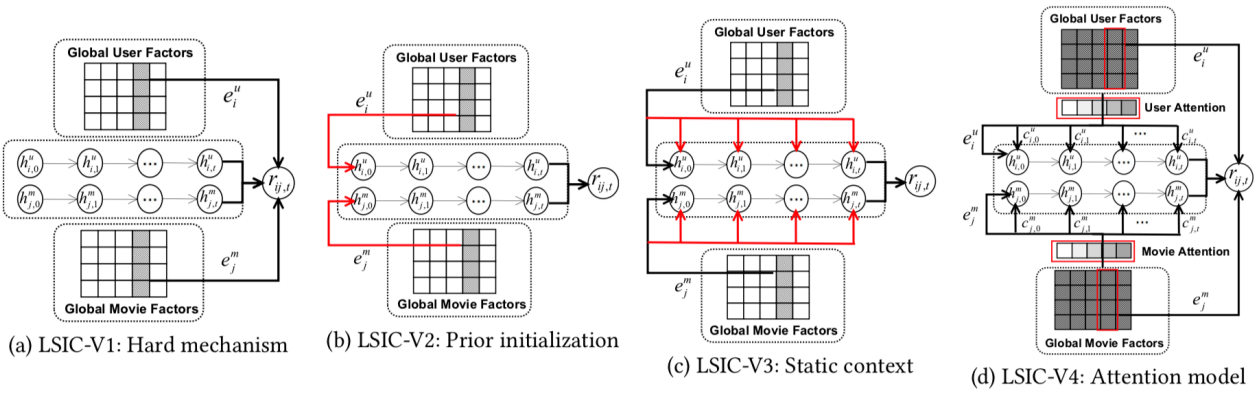
$$h_{i,t}^u = LSTM(h_{i,t-1}^u, z_{i,t}^u)$$

$$h_{j,t}^m = LSTM(h_{j,t-1}^m, z_{j,t}^m)$$

这里的 z 是 LSTM 在 t 时刻的输入，具体来说，采用 one-hot 编码方式，$z_{i,t}^u \in \mathbb{R}^U, z_{j,t}^m \in \mathbb{R}^M$。由于本文还融合了电影的海报信息，体现在设置 $z_{j,0}^m = CNN(P_j)$，通过 CNN 提取海报图片的特征信息。

综合方式，MF + RNN

简单来说，就是构造一个映射函数 g，令 $r_{ij,t} = g(e^u, e^m, h_{i,t}^u, h_{j,t}^m)$，本文提出了四种方式来构建：



(a) LSIC-V1: Hard mechanism (b) LSIC-V2: Prior initialization (c) LSIC-V3: Static context (d) LSIC-V4: Attention model

Figure 2: Four strategies to calculate the score function g, integrating MF and RNN.

1. LSIC-V1，一种 hard 机制融合，

$$r_{ij,t} = g(e^u, e^m, h^u_{i,t}, h^m_{j,t}) = \frac{1}{1+\exp(-s)}$$

这里

$$s = e^u_i \cdot e^m_j + h^u_{i,t} \cdot h^m_{j,t} + b^u_i + b^m_j$$

其中，b 表示用户和电影的打分偏置。事实上，这种处理方式并没有使用全局的信息来指导局部（当前序列的特征变换）的学习，比较生硬。

2. LSIC-V2，从 MF 中得到的 e^u_i 和 e^m_j 作为 LSTM 的初始化状态。
3. LSIC-V3，除了把 e^u_i 和 e^m_j 作为初始状态，还作为额外的输入信息加到 LSTM 的输入层上。
4. LSIC-V4，加入 attention 机制，$r_{ij,t}$ 的计算还考虑到了 attention 向量的影响，增加对应的上下文信息 $c^u_{i,t}$ 和 $c^m_{j,t}$，具体的操作和机器翻译、情感分析中的 Vanilla Attention 一致。

GAN 与推荐系统结合

GAN 包含一个生成器 G 和判别器 D，是一个经典的极小化极大值的过程。本文的模型中，迭代优化生成器和判别器，G 负责根据历史数据生成推荐列表，D 负责评价生成的结果和标准结果的相关性，换句话说，D 要尽量区分 G 生成的结果和给定的结果，而 G 要尽可能迷惑 D 使其难以分辨，从而达到 G 能够生成足够合理、正确的待推荐列表。

1. 判别器模型

图 1 中的右半边即是一个判别器 D。可以看到，通过一个对称的孪生网络，即共享底层参数的方式，计算 pair-wise 损失来优化参数。判别器 D 的目标是在给定生成器的情况下，最大化能区分生成器的结果和标准结果的概率：

$$\theta^* = \arg\max_\theta \sum_{i \in \mathcal{U}} \Big(\mathbb{E}_{m_+, m_- \sim p_{true}} [\log D_\theta(u_i, m_-, m_+ | t)] + \mathbb{E}_{m_+ \sim p_{true}, m_{g,t} \sim G_\phi(m_{g,t} | u_i, t)} \left[\log(1 - D_\theta(u_i, m_{g,t}, m_+ | t) \right] \Big) \tag{20}$$

这里 m_+ 和 m_- 分别是正样本和负样本，这个损失其实是 Hinge Loss，在排序损失上通常具有较好的表现，可以视作强化学习中的 reward：

$$D(u_i, m_-, m_+|t) = \max\left\{0, \epsilon - g(\mathbf{e}_i^u, \mathbf{e}_{m_+}^m, \mathbf{h}_{i,t}^u, \mathbf{h}_{m_+,t}^m) + g(\mathbf{e}_i^u, \mathbf{e}_{m_-}^m, \mathbf{h}_{i,t}^u, \mathbf{h}_{m_-,t}^m)\right\} \tag{21}$$

2. 生成器模型

   当判别器固定是，生成器 G 在给定了用户 u_i 和当前时刻 t 的情况下，最小化这样一个目标：

$$\phi^* = \underset{\phi}{\operatorname{argmin}} \sum_{m \in \mathcal{M}} \left(\mathbb{E}_{m_{g,t} \sim G_\phi(m_{g,t}|u_i,t)}[\log(1 - D(u_i, m_{g,t}, m_+|t)]\right) \tag{22}$$

   实际操作的时候，等价于最大化 $\log D$ 这一项。

3. 梯度策略，Policy Gradient

   从生成器 G 给出的待推荐列表中的采样 m_- 的过程是离散的，所以不能直接使用 GAN 的梯度下降策略更新参数。本文采用强化学习中的 policy gradient 策略来优化 G，使之能够生成具有更高 reward 的待推荐列表，推导如下，没有很懂它具体的推导过程：

$$\begin{aligned}
\nabla_\phi J^G(u_i) &= \nabla_\phi \mathbb{E}_{m_{g,t} \sim G_\phi(m_{g,t}|u_i,t)}[\log D(u_i, m_{g,t}, m_+|t)] \\
&= \sum_{m \in \mathcal{M}} \nabla_\phi G_\phi(m|u_i,t) \log D(u_i, m, m_+|t) \\
&= \sum_{m \in \mathcal{M}} G_\phi(m|u_i,t) \nabla_\phi \log G_\phi(m|u_i,t) \log D(u_i, m, m_+|t) \\
&= \mathbb{E}_{m_{g,t} \sim G_\phi(m|u_i,t)}[\nabla_\phi \log G_\phi(m_{g,t}|u_i,t) \log D(u_i, m_{g,t}, m_+|t)] \\
&\approx \frac{1}{K} \sum_{k=1}^{K} \nabla_\phi \log G_\phi(m_k|u_i,t) \log D(u_i, m_k, m_+|t)
\end{aligned} \tag{23}$$

   GAN 的完整算法流程如下：

Algorithm 1: Long and Session-based Ranking Model with Adversarial Network

Input: generator G_ϕ, discriminator D_θ, training data S.
Initialize models G_ϕ and D_θ with random weights, and pre-train them on training data S.
repeat
for *g-steps* **do**
  Generate recommendation list for user i at time t using the generator G_ϕ.
  Sample K candidates from recommendation list.
  for $k \in \{1, ..., K\}$ **do**
    Sample a positive movie m_+ from S.
    Compute the reward $\log D(u_i, m_k, m_+|t)$ with Eq.(21)
  Update generator G_ϕ via policy gradient Eq.(23).
for *d-steps* **do**
  Use current G_ϕ to generate a negative movie and combined with a positive movie sampled from S.
  Update discriminator D_θ with Eq.(20).
until convergence

实验设置

为了验证 LSIC 模型的有效性，在 Movielens-100K 和 Netflix 的子集和全集上进行了评价。把数据集中的某个时间点之前的记录都作为训练数据，之后的作为验证数据，将其中的 5 星、4星打分视作正面数据，其他的都是未知（负面）数据。

实验实现的细节上，其他的对比实验都使用了默认的超参（难道不应该微调一下使其达到在该数据集上的最好结果才不失公正吗？）具体的设置可以参见原文 4.2、4.3 节的内容。评价标准采用 Precision@N 和 NDCG@N，以及 MAP （Mean Average Precision）和 MRR（Mean Reciprocal Ranking）。

对比的基线实验有：

- BPR，根据给定的一个正样本，最大化与平均采样的负样本之间的距离，是 top-N 推荐系统的常用 baseline；
- PRFM（Pair-wise Ranking Factorization Machine），由成对分类损失调整分解模型，最初是用来做微博推荐的；
- LambdaFM，直接采用基于排序的损失来优化分解模型的参数，使用开源代码的默认参数设置；
- RRN，具体内容可参见之前写的论文解读，相当于一种硬融合机制；
- IRGAN，原本用作信息检索，这里改造成推荐检索的任务。

实验结果

数值指标对比

与各种基线实验对比，取得了最好的效果，以 LSIC-V4 为最佳，得到了超过 5% 的提升，在其他数据集上也是类似的。

Table 3: Moive recommendation results (MovieLens).

	Precision@3	Precision@5	Precision@10	NDCG@3	NDCG@5	NDCG@10	MRR	MAP
BPR	0.2795	0.2664	0.2301	0.2910	0.2761	0.2550	0.4324	0.3549
PRFM	0.2884	0.2699	0.2481	0.2937	0.2894	0.2676	0.4484	0.3885
LambdaFM	0.3108	0.2953	0.2612	0.3302	0.3117	0.2795	0.4611	0.4014
RRN	0.2893	0.2740	0.2480	0.2951	0.2814	0.2513	0.4320	0.3631
IRGAN	0.3022	0.2885	0.2582	0.3285	0.3032	0.2678	0.4515	0.3744
LSIC-V1	0.2946	0.2713	0.2471	0.2905	0.2801	0.2644	0.4595	0.4066
LSIC-V2	0.3004	0.2843	0.2567	0.3122	0.2951	0.2814	0.4624	0.4101
LSIC-V3	0.3105	0.3023	0.2610	0.3217	0.3086	0.2912	0.4732	0.4163
LSIC-V4	**0.3327**	**0.3173**	**0.2847**	**0.3512**	**0.3331**	**0.2939**	**0.4832**	**0.4321**
Impv	7.05%	7.45%	9.00%	6.36%	6.87%	5.15%	4.79%	7.65%

为了解释对抗网络的训练过程，从下图中的学习曲线中，可以看到生成器、判别器在训练过程中，前者生成的样本对于判别器而言，越来越具有迷惑性，即生成了更为合理的推荐结果。

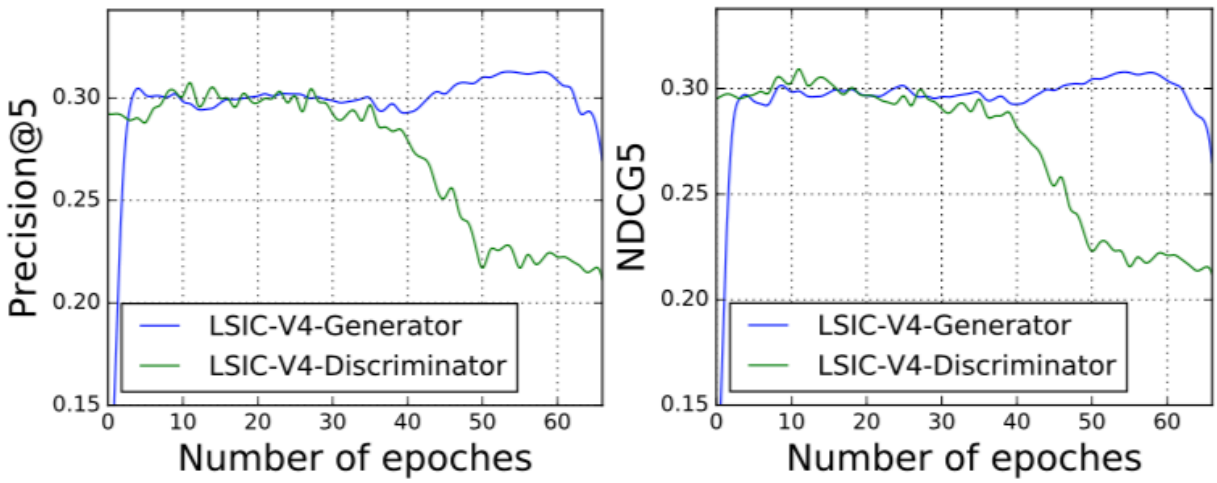


Figure 3: Learning curves of GAN on Netflix-3M

控制变量测试（Ablation Study）

作者针对 LSIC-V4 模型，采取了控制变量法，移除了某些模块进行了对比实验。如果抛弃强化学习根据奖励的梯度更新策略，对抗网络将失去作用。此外，如果没有融合辅助信息——电影的海报图片特征，性能也会下降。

Table 6: Ablation results for Netflix-3M dataset.

	Precision@3	Precision@5	Precision@10	NDCG@3	NDCG@5	NDCG@10	MRR	MAP
LSIC-V4	**0.3221**	**0.3193**	**0.2921**	**0.3157**	**0.3114**	**0.2975**	**0.4501**	**0.4247**
w/o RL	0.3012	0.2970	0.2782	0.2988	0.2927	0.2728	0.4431	0.4112
w/o poster	0.3110	0.3012	0.2894	0.3015	0.3085	0.2817	0.4373	0.4005

Case Study

作者为了进一步说明该模型的好，通过两个用户的实例测试展示了一些推荐结果，各种不同算法给出的不同的推荐列表，而 LSIC-V4 由于考虑的长期、短期时间内的特征和辅助特征，推荐的结果和用户的实际选择最为接近。

Re-rank 效果

由于对整个电影数据进行全局的排序是十分费时的，首先使用 MF 方法根据得分选出前若干个候选，然后再由 LSIC-V4 模型进行重排序得到结果，得以大大减小排序的空间和时间代价。实验表示，选择 100 个是较为合理的。

序列长度的敏感性分析

本文还关注到了对于序列时间跨度预测结果的不同，由于引入了 RNN 将序列信息融合在推荐模型中，较大地提高了中等时间跨度的序列化推荐的性能。下图中红色部分是 LSIC-V4 模型带来的提升，可见那些较短的时间跨度的序列提升不明显，因为在这个短期中体现的动态特征不具有十分明显的变化，而且存在很多用户在一定时期内爆发式的评论，这个时序信息的质量不高。（这个解释是我认为的，作者没有解释）

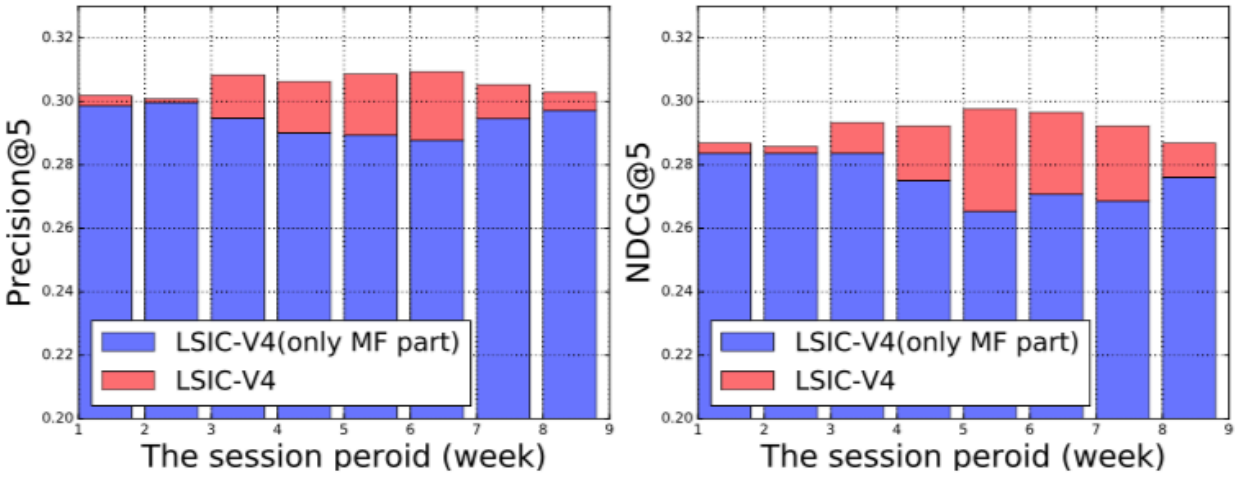


Figure 5: Sensitivity of the session period on Netflix-3M

结论与点评

本文将传统做法 MF 和现下比较热门的用 RNN 模型处理序列化信息，进行序列化推荐的研究方向融合在一起，有理有据地说明了 MF 提取的是长期的特征，RNN 处理动态特征上的合理性，通过多种融合方式（LSIC-V1~V4），常规地加上 Attention，并且引入了外部信息来缓解冷启动。还把刚刚火起来的强化学习、GAN 用在一起，可以认为是把各种能用的武器都利用起来，感觉像个大杂烩，通篇读完觉得有一些道理，但是动机这块解释的比较欠缺，基本都是从相关工作中的动机直接拿来用的。

由于本文涉及到的技术过多，我认为是一项较为复杂且成功的项目，文章能解释 what & how，但是 why 方面有些欠缺，让人觉得有点道理，但不够信服，虽然最后给出的 case study、控制变量、对比实验等有一些体现，但是好像分析的不够透彻。目前本文似乎还没有公开发表出来，作者承诺将在文章发表后公开代码，可以期待一下，对于我进行相关研究是很好的代码参考。

我认为，本文融合了太多的东西，模型中使用两个 RNN 来分别处理用户和电影的特征，还用的是 one-hot 这样的稀疏编码，难以体现用户、电影之间的潜在关系，可以考虑加一层 embedding。我之前的想法是，用一个 RNN 处理输入序列，t 时刻用户 i 的特征和电影 j 的特征都使用 embedding 作为长期、静态的特征输入，而动态特征则交给 RNN 每个时刻的隐状态 h_t，可以精简模型的设计和运行开

销，并且相比一些传统的做法有一些微小的提升。也考虑过 MF 作为 Fine-tuned 的初始化这样的融合方式、加入 Attention 机制这样的通常操作，和作者的诸多想法不谋而合。不过这里的 GAN + RL 还是比较有意思的，虽然设计还是有些简单，但根据推荐系统的实际任务做出了不少的修改，不得不说 RL 中 reward 的概念和生成候选推荐的质量是有很强的关联的。

最后，我认为本文的 future work 可以是设计更为合理的 RL 模型来解决 Session-based Recommendation 问题。