自助式顺序推荐

Wang-Cheng Kang, Julian McAuley
UC San Diego
{wckang,jmcauley}@ucsd.edu

摘要--
连续动态是许多现代推荐系统的一个关键特征，这些系统试图根据用户最近的行为来捕捉他们活动的"文本"。为了捕捉这种模式，有两种方法得到了推广。马尔科夫链（MCs）和循环神经网络（RNNs）。马尔科夫链假设用户的下一个动作可以根据他们最后（或最后几个）的动作来预测，而RNN原则上允许发现更长期的语义。一般来说，基于MC的方法在极其稀疏的数据集中表现最好，在这种情况下，模型的简洁性至关重要，而RNN在密集的数据集中表现更好，在这种情况下，更高的模型复杂性是可以承受的。我们的工作目标是平衡这两个目标，提出了一个基于自我注意的顺序模型（SASRec），它允许我们捕捉长期语义（像RNN），但是，使用注意机制，根据相对较少的行动（像MC）进行预测。在每一个时间步骤中，SASRec试图识别哪些项目从用户的行动历史中找出"相关的"，并利用它们来预测下一个项目。广泛的实证研究表明，我们的方法在稀疏和密集的数据集上都优于各种最先进的顺序模型（包括基于MC/CNN/RNN的方法）。此外，该模型比类似的基于CNN/RNN的模型要高效一个数量级。注意力权重的可视化也显示了我们的模型如何自适应地处理各种密度的数据集，并在活动序列中发现有意义的模式。

I. 简介

顺序推荐系统的目标是将用户行为的个性化模型（基于历史活动）与基于用户最近行为的某种"背景"概念结合起来。从序列动态中捕捉有用的模式是具有挑战性的，主要是因为输入空间的维度随着作为背景的过去行为的数量而呈指数级增长。因此，顺序推荐的研究主要关注的是如何简洁地捕捉这些高阶动态。

马尔科夫链（MCs）是一个典型的例子，它假设下一个行动只取决于之前的行动（或之前的几个），并且已经成功地被采用来描述短距离的项目转换以进行推荐[1]。另一项工作是使用循环神经网络（RNN），通过一个隐藏的状态来总结以前的所有行动，用来预测下一个行动[2]。

这两种方法虽然在特定情况下都很强大，但在某种程度上都仅限于某些类型的数据。基于MC的方法，通过强有力的简化假设，在高稀疏度环境中表现良好，但可能无法捕捉到更复杂场景的复杂动态。相反，RNNs，虽然表现力强，但需要大量的数据（特别是*密集的*数据），然后才能超越更简单的基线。

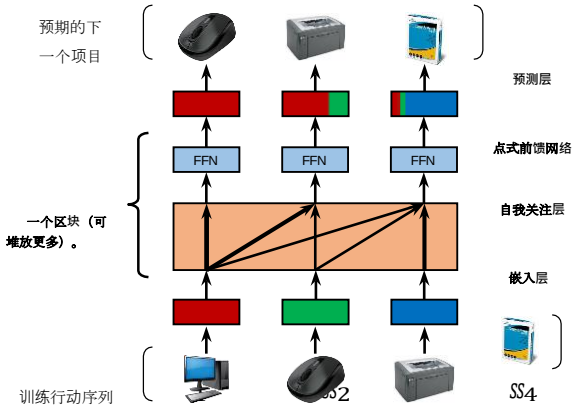


图1：显示SASRec训练过程的简化图。在每一个时间步骤中，模型考虑所有以前的项目，并使用注意力来"关注"与下一个行动相关的项目。

最近，一个新的顺序模型*Transfomer*在机器翻译任务中取得了最先进的性能和效率[3]。与现有的使用卷积或递归模块的顺序模型不同，Transformer纯粹是基于一种被称为"自我注意"的注意力机制，它具有很高的效率，能够发现句子中单词之间的句法和语义模式。

受这种方法的启发，我们试图将自我关注机制应用于顺序推荐问题。我们希望这个想法能够解决上述两个问题，一方面能够从过去的所有行动中汲取背景信息（如RNNs），但另一方面能够仅以少数行动为框架进行预测（如MCs）。具体来说，我们建立了一个基于自我关注的顺序推荐模型（*SASRec*），它在每个时间步骤中都会自适应地给以前的项目分配权重（图1）。

所提出的模型在几个基准数据集上的表现明显优于最先进的基于MC/CNN/RNN的顺序推荐方法。特别是，我们考察了作为数据集稀疏度函数的性能，其中模型性能与上述模式密切相关。由于自我关注机制，SASRec倾向于考虑密集数据集上的长距离依赖关系，而关注稀疏数据集上的更多近期活动。这被证明是适应性地处理具有不同密度的数据集的关键。

此外，核心部分（即自我注意

块）的SASRec适合于并行加速，从而使模型比基于CNN/RNN的替代品快一个数量级。此外，我们还分析了SASRec的复杂性和可扩展性，进行了全面的消融研究以显示关键组件的效果，并将注意力权重可视化以定性地揭示模型的行为。

II. 相关工作

有几项工作与我们密切相关。我们首先讨论一般推荐，然后是时间性推荐，最后讨论顺序性推荐（特别是MC和RNN）。最后我们介绍注意力机制，特别是作为我们模型核心的自我注意力模块。

A. 一般建议

推荐系统的重点是根据历史反馈（如点击、购买、喜欢），对用户和物品之间的兼容性进行建模。用户反馈可以是*显性的*（如评级）或*隐性的*（如点击、购买、评论）[4], [5]。由于解释"非观察"（如非购买）数据的模糊性，对隐性反馈进行建模可能具有挑战性。为了解决这个问题，人们提出了*点式*[4]和*对式*[5]方法来解决这种挑战。

矩阵分解(MF)方法试图发现潜在的维度来代表用户的偏好和项目的属性，并通过用户和项目嵌入之间的内积来估计相互作用[6], [7]。此外，另一条工作路线是基于项目相似度模型（ISM），并不明确地用潜伏因素来模拟每个用户（例如FISM[8]）。他们学习一个项目对项目的相似性矩阵，并通过测量用户与之前互动过的项目的相似性来估计用户对某个项目的偏好。

最近，由于在相关问题上的成功，各种深度学习技术已经被引入到推荐领域[9]。其中一个工作方向是使用神经网络来提取项目特征（如图像[10]，[11]，文本[12]，[13]等），用于内容感知的推荐。另一条线的工作是试图取代传统的MF。例如，NeuMF[14]通过多层感知（MLP）估计用户的偏好，而AutoRec[15]使用自动编码器预测评级。

B. 时间性建议

追溯到*Netflix*奖，时间性推荐通过明确地对用户活动的时间戳进行建模，在各种任务上显示出强大的性能。TimeSVD++[16]通过将时间分割成几个片段，并在每个片段中分别对用户和项目进行建模，取得了强大的效果。这样的模型对于理解那些表现出明显的（短期或长期）时间"漂移"的数据集（例如，"在过去10年中，电影偏好是如何变化的"，或者"用户在下午4点会访问什么样的企业"，等等）至关重要[16]-[18]。顺序推荐（或下一个项目推荐）与这种设定略有不同，因为它只考虑行动的顺序，并建立独立于时间的顺序模式。从本质上讲，顺序模型试图根据用户最近的活动来模拟他们行动的 "背景"，而不是考虑时间模式*本身*。

C. 依次推荐

许多顺序推荐系统试图对项目-项目过渡矩阵进行建模，作为捕捉连续项目间顺序模式的一种手段。例如，FPMC融合了一个MF项和一个项目-项目过渡项来分别捕捉长期偏好和短期过渡[1]。从本质上讲，捕获的过渡是一个一阶马尔科夫链（MC），而高阶MC则假定下一个行动与之前的几个行动有关。由于最后访问的项目往往是影响用户下一步行动的关键因素（本质上是提供"上下文"），基于一阶MC的方法显示出强大的性能，尤其是在稀疏的数据集上[19]。还有一些采用高阶MC的方法，考虑了更多以前的项目[20], [21]。特别是卷积序列嵌入（Caser），一种基于CNN的方法，将L个先前项目的嵌入矩阵视为一个'图像'，并应用卷积运算来提取过渡[22]。

除了基于MC的方法外，另一个工作方向是采用RNNs对用户序列进行建模[2], [23]-[25]。例如，GRU4Rec使用门控递归单元（GRU）对点击序列进行建模，用于基于会话的推荐[2]，而改进后的版本进一步提高了其Top-N推荐性能[26]。在每个时间步骤中，RNN将上一个步骤的状态和当前的行动作为其输入。这些依赖性使得RNN的效率降低，尽管像"会话并行"这样的技术已经被提出来以提高效率[2]。

D. 注意机制

注意机制已被证明在各种任务中是有效的，如图像说明[27]和机器翻译[28]等。从本质上讲，这种机制背后的想法是，连续的输出（例如）都取决于模型应该连续关注的一些输入的"相关"部分。另一个好处是，基于注意力的方法通常更容易解释。最近，注意力机制已经被纳入到推荐系统中[29]-[31]。例如，注意力因素化机器（AFM）[30]学习每个特征交互的重要性，用于内容感知的推荐。

然而，上述所使用的注意力技术本质上是原始模型的一个*附加*组件（如注意力+RNNs，注意力+FMs，等等）。最近，一种纯粹基于注意力的序列到序列的方法，Transfomer[3]，在机器翻译任务上取得了最先进的性能和效率，这些任务以前一直由基于RNN/CNN的方法主导[32], [33]。Transformer模型在很大程度上依赖于所提出的"自我注意"模块来捕捉句子中的复杂结构，并检索相关的词（在源语言中）以生成下一个词（在目标语言中）。受Transformer的启发，我们试图在自我注意方法的基础上建立一个新的顺序推荐模型，尽管顺序推荐的问题与机器翻译完全不同，需要专门设计的模型。

表一。符号。

符号	说明
U, I	用户和项目集
S^u	用户u的历史互动序列。
	$(S_1^u, S_2^u, ..., S_{\|S^u\|}^u)$
$d \in \mathbb{N}$	潜在向量的维度
$n \in \mathbb{N}$	最大序列长度
$b \in \mathbb{N}$	自我注意区块的数量
$\mathbf{M} \in \mathbb{R}^{\|I\| \times d}$	项目嵌入矩阵
$\mathbf{P} \in \mathbb{R}^{n \times d}$	位置嵌入矩阵
$\mathbf{E}_{\square} \in \mathbb{R}^{n \times d}$	输入嵌入矩阵
$\mathbf{S}^{(b)} \in \mathbb{R}^{n \times d}$	b-th自我注意层后的项目嵌入
$\mathbf{F}^{(b)} \in \mathbb{R}^{n \times d}$	b-th 前馈网络后的项目嵌入

III. 方法论

在顺序推荐的设定中，我们得到一个用户的行动序列$S^u = (S_1^u, S_2^u, ..., S_{|S^u|}^u)$，并寻求预测下一个项目。在训练过程中，在时间步骤$t$，模型根据之前的$t$个项目来预测下一个项目。如图1所示，我们可以把模型的输入看作$(S_1^u, S_2^u, ..., S_{|S^u|-1}^u)$，其预期输出为同一序列的 "移位 "版本。$(S_2^u, S_3^u, ..., S_{|S^u|}^u)$。在本节中，我们将描述我们如何建立一个通过嵌入层、几个自我关注块和一个预测层的顺序推荐模型。我们还分析了其复杂性，并进一步讨论了SASRec与相关模型的不同之处。我们的符号总结在表一中。

A. 嵌入层

我们将训练序列（$S_1^u, S_2^u, ..., S_{|S^u|-1}^u$）的成一个固定长度的序列$s = (s_1, s_2, ..., s_n)$，其中$n$代表我们的模型可以处理的最大长度。如果序列的长度大于n，我们就考虑最近的n个行动。如果序列的长度小于n，我们就在左边重复添加一个 "填充 "项，直到长度为n.我们创建一个物品嵌入矩阵$\mathbf{M} \in \mathbb{R}^{|I| \times d}$其中$d$是潜伏维数，并检索出输入嵌入矩阵$\mathbf{E} \in \mathbb{R}^{n \times d}$，其中$\mathbf{E}_i = \mathbf{M}_s$。一个恒定的零向量$\mathbf{o}$被用来作为填充项的嵌入。

位置嵌入。正如我们在下一节所看到的，由于自我注意模型不包括任何递归或卷积模块，它不知道以前的项目的位置。因此，我们注入一个可学习的位置嵌入$\mathbf{P} \in \mathbb{R}^{n \times d}$到输入嵌入。

$$\mathbf{E}_{\square} = \begin{bmatrix} \mathbf{M}_{s_1} + \mathbf{P}_1 \\ \mathbf{M}_{s_2} + \mathbf{P}_2 \\ \dots \\ \mathbf{M}_{s_n} + \mathbf{P}_n \end{bmatrix} \quad (1)$$

我们还尝试了[3]中使用的固定位置嵌入，但发现在我们的案例中，这导致了更差的性能。我们在我们的实验中，定量和定性地分析了位置嵌入的效果。

B. 自我关注块

缩放点积注意[3]被定义为。

$$\text{注意}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right)\mathbf{V}, \quad (2)$$

其中$\mathbf{Q}$代表查询，$\mathbf{K}$代表键，$\mathbf{V}$代表值。(每一行代表一个项目)。直观地说，注意力层计算所有数值的加权和，其中权重为询问i和价值j之间的互动关系√与比例系数d是为了避免内积的数值过大，特别是在维度较高的情况下。

自我注意层。在机器翻译等NLP任务中，注意力机制通常使用$\mathbf{K}=\mathbf{V}$（例如，使用RNN编码器-解码器进行翻译：编码器的隐藏状态是键和值，解码器的隐藏状态是查询）[28]。最近，有人提出了一种自我关注的方法，它使用相同的对象作为查询、键和值[3]。在我们的案例中，自我注意操作将嵌入$\mathbf{E}$作为输入，通过线性投影将其转换为三个矩阵，并将其送入注意层。

$$\mathbf{S} = \text{SA}(\mathbf{E}) = \text{Attention}(\mathbf{E}\mathbf{W}^Q, \mathbf{E}\mathbf{W}^K, \mathbf{E}\mathbf{W}^V)。 \quad (3)$$

其中投影矩阵$\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V \in \mathbb{R}^{d \times d}$。

投射使模型更加灵活。例如，该模型可以学习不对称的相互作用（即<查询i，键j>和<查询j，键i>可以有不同的相互作用）。

因果关系。由于序列的性质，模型在预测(t+1)第1个项目时应该只考虑前t个项目。然而，自我注意层的第t个输出($\mathbf{S}_t$)包含后续项目的嵌入，这使得的模型不合适。因此，我们修改注意力，禁止$\mathbf{Q}_i$和$\mathbf{K}_j$（$j > i$）之间的所有链接。

Point-wise Feed-Forward网络。尽管自我关注能够以自适应的权重聚合所有以前的项目嵌入，但最终它仍然是一个线性模型。为了赋予该模型以非线性，并考虑不同潜在维度之间的相互作用，我们对所有的$\mathbf{S}_i$（共享参数）应用一个点式两层前馈网络。

$$\mathbf{F}_i = \text{FFN}(\mathbf{S}_i) = \text{ReLU}(\mathbf{S}_i\mathbf{W}_i^{(1)} + \mathbf{b}^{(1)})\mathbf{W}^{(2)} + \mathbf{b}^{(2)}, \quad (4)$$

其中$\mathbf{W}^{(1)}$，$\mathbf{W}^{(2)}$是$d \times d$矩阵，$\mathbf{b}^{(1)}$，$\mathbf{b}^{(2)}$是d维向量。请注意，$\mathbf{S}_i$和$\mathbf{S}_j$（$i = j$）之间没有互动，这意味着我们仍然可以防止信息泄露（从后到前）。

C. 叠加自留地块

在第一个自我关注区块之后，$\mathbf{F}_i$基本上聚合了之前所有项目的嵌入（即$\mathbf{E}_{j \le i}$），可能有助于通过另一个基于$\mathbf{F}$的自我注意块来学习更复杂的项目转换。具体来说，我们把自我注意区块（即自我注意层和前馈网络），第b（b>1）区块定义为：。

$$\mathbf{S}_i^{(b)} = \text{SA}(\mathbf{F}^{(b-1)})。 \quad \mathbf{F}^{(b)}$$

$$= \mathrm{FFN}(\mathbf{S}^{(b)}), \forall i \in \{1, 2, \ldots, n\}, \tag{5}$$

和1-st块定义为$\mathbf{S}^{(1)} = \mathbf{S}$和$\mathbf{F}^{(1)} = \mathbf{F}$。

然而，当网络深入时，有几个问题会变得更加严重。1）模型容量的增加导致过度拟合；2）训练过程变得不稳定（由于梯度消失等原因）；3）具有更多参数的模型往往需要更多的训练时间。受[3]的启发，我们进行了以下操作来缓解这些问题。

$$g(x) = x + \text{Dropout}(g(\text{LayerNorm}(x)))。$$

其中$g(x)$代表自我注意层或前馈网络。也就是说，对于每个区块中的层g，我们在输入到g之前对输入x进行层规范化，对g的输出应用dropout，并将输入x加入到最终输出。我们在下面介绍这些操作。

剩余连接。在某些情况下，多层神经网络已经显示出学习有意义的能力。

层次化的特征[34]。然而，简单地增加更多
层不容易对应更好的性能，直到剩余网络被提出[35]。其核心思想是
残差网络是通过残差连接将低层特征传播到高层。因此，如果低层特征是有用的，该模型可以很容易地将它们传播到最后一层。同样地，我们假设残差连接在我们的案例中也是有用的。例如，现有的顺序推荐方法表明，最后访问的项目对预测下一个项目起着关键作用[1]，[19]，[21]。然而，在几个自我关注块之后，最后访问过的项目的嵌入与之前的所有项目纠缠在一起；增加剩余连接，将最后访问过的项目的嵌入传播到最后一层，将使模型更容易利用低层信息。

层归一化。层归一化用于归一化各特征的输入（即零均值和单位方差），这对稳定和加速神经网络训练是有益的[36]。与批量归一化[37]不同，层归一化中使用的统计数据与同一批次中的其他样本无关。具体来说，假设输入是一个包含样本所有特征的向量$\mathbf{x}$，该操作被定义为。

$$\text{LayerNorm}(\mathbf{x}) = \boldsymbol{\alpha} \odot \frac{\mathbf{x} - \mu}{\sqrt{\sigma^2 + \varepsilon}} + \boldsymbol{\beta}。$$

其中，$\odot$是一个元素的乘积（即Hadamard乘积），μ和σ是$\mathbf{x}$的平均值和方差，$\boldsymbol{\alpha}$和$\boldsymbol{\beta}$是学习的缩放因子和偏置项。

辍学。为了缓解深度神经系统中的过拟合问题
网络，"辍学 "正则化技术已被证明是

在各种神经网络架构中是有效的[38]。辍学的想法很简单：在训练期间以概率p随机 "关闭"神经元，并在训练结束后使用所有神经元。
检验。进一步的分析指出，辍学可以看作是
作为一种集合学习的形式，它考虑了大量的模型（在神经元数量上是指数级的，而且是在一个人身上）。

输入特征），共享参数[39]。我们还在嵌入的$\mathbf{E}_i$上应用了一个剔除层。

D. 预测层

在自适应地、分层次地提取以前消费过的物品的信息的b个自我注意块之后，我们根据$\mathbf{F}^{(b)}$，预测下一个物品（给定前t个物品）。具体来说，我们采用一个MF层来预测物品i的相关性。

$$r_{i,t} = \mathbf{F}_t^{(b)} \mathbf{N}_i^T$$

其中，$r_{i,t}$是指在前t个项目（即s_1，s_2，...，s_t）的情况下，项目i是下一个项目的相关性，$\mathbf{N} \in \mathbb{R}^{|I| \times d}$是一个项目嵌入矩阵。因此，一个高的交互得分$r_{i,t}$意味着高度的相关性，我们可以通过对分数的排名来产生推荐。

共享的项目嵌入。为了减少模型的大小和缓解过度拟合，我们考虑另一种方案，它只使用一个单一的项目嵌入$\mathbf{M}$。

$$r_{i,t} = \mathbf{F}_t^{(b)} \mathbf{M}_i^T。 \tag{6}$$

请注意，$\mathbf{F}_t^{(b)}$可以表示为一个取决于项目嵌入$\mathbf{M}$的函数：$\mathbf{F}_t^{(b)} = f(\mathbf{M}_{s_1}, \mathbf{M}_{s_2}, ..., \mathbf{M}_{s_t})$。
使用同质物品嵌入的一个潜在问题是，它们的内部产品不能代表不对称的物品转换（例如，物品i经常在j之后被购买，但反之则不然），因此现有的方法如FPMC倾向于使用异质的物品嵌入。然而，我们的模型没有这个问题，因为它学习了一个非线性的转换。例如，前馈网络可以在相同的项目嵌入下轻松实现不对称。ffn（$\mathbf{m}_i$）$\mathbf{m}_j^T$ = ffn（$\mathbf{m}_j$）$\mathbf{m}_i^T$。从经验上看，使用共享的项目嵌入可以显著提高我们模型的性能。

明确的用户建模。为了提供个性化的建议，现有的方法通常采取两种方法之一。
1）学习代表用户偏好的*显式*用户嵌入（例如MF[40]、FPMC[1]和Caser[22]）；2）考虑用户以前的行为，并从访问过的项目的嵌入中诱导出一个*隐式*用户嵌入（例如FSIM[8]、Fossil[21]、GRU4Rec[2]）。我们的方法属于后者，因为我们通过考虑一个用户的所有行动来生成一个嵌入$\mathbf{F}^{(b)}$。然而，我们也可以在以下位置插入一个明确的用户嵌入最后一层，例如通过加法：$r_{u,i,t} = (\mathbf{U}_u + \mathbf{F}^{(b)})\mathbf{M}_i^T$其中$\mathbf{U}$是一个用户嵌入矩阵。然而，我们根据经验发现，增加一个明确的用户嵌入并不能提高性能（可能是因为模型已经考虑了用户的所有行为）。

E. 网络培训

回顾一下，我们把每个用户序列（不包括最后一个动作）$(S_u^1, S_u^2, ..., S_u^{|S_u|-1})$转换成一个固定长度的序列$s = s_1, s_2, ..., s_n$通过截断或填充项目。我们定义$o_t$作为时间步骤$t$的预期产出。

$$o_t = \begin{cases} \text{<pad>}, & \text{如果 } s_t \text{ 是一个填充项} \\ S_u^{t+1}, & 1 \le t < |S_u| \\ S_u^{|S_u|}, & t = n \end{cases},$$

其中<pad>表示一个填充项。我们的模型将一个序列s作为输入，将相应的序列o作为预期的输入。

输出，并且我们采用二元交叉熵损失作为目标函数。

$$-\sum_{S^u \in S}\sum_{t\in[1,2,\ldots,n]}\left[\log(\sigma(r_{o\,t,t}))+\sum_{j\notin S}\log(1-\sigma(r_{j,t}))\right]。$$

注意，我们忽略了o_t = <pad>的条款。

网络由*Adam*优化器[41]优化，它是随机梯度下降（SGD）的一个变种，具有自适应矩估计。在每个历时中，我们为每个序列中的每个时间步骤随机生成一个负项j。更多的实施细节将在后面描述。

F. 复杂度分析

空间复杂度。我们模型中的学习参数来自于嵌入和自我注意中的参数。

层、前馈网络和层规范化。参数总数为$O(|I|d + nd + d^2)$，与其他方法相比是适中的（例如，$O(|U|d + |I|d)$为FPMC），因为它不随用户数量的增长而增长，而且在推荐问题中，d通常很小。

时间复杂度。我们的模型的计算复杂度主要是由自我注意层和前馈网络造成的，它是$O(n^2 d + nd^2)$。主导项通常是来自自我注意层的$O(n^2 d)$。然而，在我们的模型中，一个方便的属性是，每个自我注意层的计算是完全可并行的，这对GPU加速是有利的。相比之下，基于RNN的方法（如GRU4Rec[2]）对时间步骤有依赖性（即时间步骤t的计算必须等待时间步骤$t-1$的结果），这导致了顺序操作的$O(n)$时间。我们根据经验发现，我们的方法比使用GPU的基于RNN和CNN的方法快十倍以上（结果类似于[3]中的机器翻译任务），而且最大长度n可以轻松扩展到几百个，这对于现有的基准数据集来说一般是足够的。

在测试过程中，对于每个用户，在计算了嵌入$\mathbf{F}^{(b)}$，n过程与标准MF方法相同。$(O(d)$用于评估对一个项目的偏好)。

交付长序列。尽管我们的实验从经验上验证了我们方法的效率，但最终它无法扩展到非常长的序列。有几个选择是有希望在未来研究的。1）使用限制性的自我注意[42]，它只注意最近的动作而不是所有的动作，远处的动作可以在更高的层次上考虑；2）像[22]那样将长序列分割成短段。

G. 讨论

我们发现，SASRec可以被看作是一些经典CF模型的泛化。我们还从概念上讨论了我们的方法和现有方法如何处理序列建模。

对现有模式的还原。

- *因子化马尔科夫链。*FMC对一阶项目过渡矩阵进行因子化，并根据最后一个项目i预测下一个项目j。

$$P(j|i) \propto \mathbf{M}_i^T \mathbf{N}_j$$

。

如果我们将自我关注块设置为零，使用不共享的项目嵌入，并去除位置嵌入，SASRec就会还原为FMC。此外，SASRec还与Factorized Personalized Markov Chains（FPMC）[1]密切相关，后者将MF与FMC相融合以捕获分别是用户偏好和短期动态。

$$P(j|u,i) \propto [\mathbf{U}_u, \mathbf{M}_i]_j \mathbf{N}^T。$$

按照上面对FMC的还原操作，并增加一个明确的用户嵌入（通过连接），SASRec等同于FPMC。

- *因素化项目相似度模型[8]*。FISM通过考虑i和用户之前消费的物品之间的相似性来估计对i的偏好分数。

$$P(j|u) \propto \left(\frac{1}{|S_u|}\right)^{\alpha} \sum_{i\in S_u} \mathbf{M}_i \mathbf{N}_j^T$$

如果我们使用一个自我注意层（不包括前馈网络），设置统一的注意权重（即：。$\frac{1}{|S_u|}$）的项目，使用不共享的项目嵌入，以及移除位置嵌入，SASRec被简化为FISM。因此，我们的模型可以被看作是一个自适应的、分层的、连续的项目相似性模型，用于推荐下一个项目。

基于MC的推荐。马尔科夫链（MC）可以有效地捕捉局部的顺序模式，假设下一个项目只依赖于前L个项目。现有的基于MC的顺序推荐方法依赖于一阶MC（如FPMC[1]、HRM[43]、TransRec[19]）或高阶MC（如Fossil[21]、Vista[20]、Caser[22]）。第一组方法往往在稀疏的数据集上表现最好。相比之下，基于高阶MC的方法有两个局限性：（1）MC的阶数L需要在训练前指定，而不是自适应选择；（2）性能和效率不会随着阶数L的变化而变化，因此L通常很小（例如小于5）。我们的方法解决了第一个问题，因为它可以自适应地关注以前的相关项目（例如，在稀疏的数据集上只关注最后一个项目，而在密集的数据集上关注更多的项目）。此外，我们的模型基本上是以之前的n个项目为条件的，并且根据经验可以扩展到几百个项目，在训练时间适度增加的情况下表现出性能提升。

基于RNN的推荐。另一条工作路线是寻求使用RNNs来为用户动作序列建模[2], [17], [26]。RNNs一般适用于对序列进行建模，尽管最近的研究表明CNNs和自我注意力在一些序列设置中会更强[3], [44]。我们的基于自我注意力的模型可以从项目相似性模型中衍生出来，这是一个合理的用于推荐的序列建模的选择。对于RNN来说，除了它们在并行计算中的低效率（第三节F），它们的最大路径长度（从一个输入节点到相关的输出节点）是$O(n)$。相比之下，我们的模型具有$O(1)$的最大路径长度，这对学习长距离的依赖关系是有益的[45]。

IV. 实验

在这一节中，我们将介绍我们的实验设置和实证结果。我们的实验是为了回答以下研究问题。

RQ1：SASRec的表现是否优于最先进的模型，包括基于CNN/RNN的方法？

RQ2：SASRec架构中的各种组件的影响是什么？

RQ3：SASRec的训练效率和可扩展性（关于n）是什么？

RQ4：注意力权重是否能够学习与位置或物品属性相关的有意义的模式？

A. 数据集

我们在三个真实世界应用的四个数据集上评估了我们的方法。这些数据集在领域、平台和稀疏度方面有很大的不同。

- **亚马逊。**在[46]中介绍的一系列数据集，包括从*Amazon.com*抓取的大型产品评论体。*亚马逊*上的顶级产品类别被当作独立的数据集处理。我们考虑两个类别，"美容"和"游戏"。这个数据集因其高度的稀疏性和可变性而引人注目。
- **蒸汽。**我们介绍了一个新的数据集，它是从大型在线视频游戏发行平台*Steam*上抓取的。该数据集包含2,567,538个用户，15,474个游戏和7,793,069条英文评论，时间跨度为2010年10月至2018年1月。该数据集还包括可能对未来工作有用的丰富信息，如用户的游戏时间、价格信息、媒体评分、类别、开发商（等）。
- **MovieLens。**一个广泛使用的基准数据集，用于评估协作过滤算法。我们使用的版本（MovieLens-1M）包括100万个用户评分。

我们沿用了[1]、[19]、[21]中相同的预处理程序。对于所有的数据集，我们将评论或评级的存在视为隐性反馈（即用户与项目进行了互动），并使用时间戳来确定行动的顺序。我们放弃了少于5个相关行动的用户和项目。为了划分，我们把每个用户的历史序列S^u分成三个部分：（1）最近的行动$S^u_{|S^u|-1}$用于测试，(2) 第二个最近的行动$S^u_{|S^u|-2}$用于验证，以及(3)所有剩余的动作用于训练。注意，在测试期间，输入序列包含训练动作和验证动作。

数据统计见表二。我们看到，两个*亚马逊*数据集的每个用户和每个项目的行动最少（平均），*Steam*的每个项目的平均行动数很高，*MovieLens-1m*是最密集的数据集。

B. 比较方法

为了显示我们方法的有效性，我们包括三组推荐基线。第一组包括一般的推荐方法，它只考虑用户的反馈，而不考虑行动的顺序。

表二：数据集统计（预处理后）。

数据集	#users	#项目	平均行动/用户	平均行动/item	#行动
亚马逊美容	52,024	57,289	7.6	6.9	0.4M
亚马逊游戏	31,013	23,715	9.3	12.1	0.3M
蒸汽	334,730	13,047	11.0	282.5	3.7M
电影镜头-1M	6,040	3,416	163.5	289.1	1.0M

- **PopRec**。这是一个简单的基线，根据项目的受欢迎程度（即相关行动的数量）进行排名。
- **贝叶斯个性化排名（BPR）[5]**。BPR是一种从隐性反馈中学习个性化排名的经典方法。有偏见的矩阵分解被用作基础推荐器。

第二组包含了基于一阶马尔科夫链的顺序推荐方法，它考虑了最后访问的项目。

- **因数化马尔可夫链（FMC）。**一种一阶马尔可夫链方法。FMC使用两个项目嵌入对项目转移矩阵进行因子化，并仅根据最后访问的项目产生建议。
- **因式化个性化马尔可夫链（FPMC）[1]**。FPMC使用矩阵因子化和因子化一阶马尔可夫链的组合作为其推荐器，它捕捉到用户的长期偏好以及项目间的转换。
- **基于翻译的推荐（TransRec）[19]**。一种最先进的一阶顺序推荐方法，它将每个用户建模为一个翻译向量，以捕捉从当前项目到下一个项目的过渡。

最后一组包含了基于深度学习的顺序识别系统，它考虑了几个（或所有）以前访问过的项目。

- **GRU4Rec[2]**。一个开创性的方法，使用RNNs对用户的行动序列进行建模，用于基于会话的推荐。我们将每个用户的反馈序列视为一个会话。
- **GRU4Rec$^+$[26]**。GRU4Rec的改进版，它采用了不同的损失函数和采样方法。

实验结果显示，在"顶层"和"底层"两方面都有显著的性能提升。

N项建议。

- **卷积序列嵌入（Caser）[22]**。最近提出的一种基于CNN的方法，通过对L个最新项目的嵌入矩阵进行卷积运算来捕捉高阶马尔科夫链，并实现了最先进的顺序推荐性能。

由于其他顺序推荐方法（例如PRME[47]、HRM[43]、Fossil[21]）在类似的数据集上的表现优于上述基线，我们省略了与它们的对比。我们也不包括像TimeSVD++[16]和RRN[17]这样的时间性推荐方法，它们的设置与我们在这里考虑的不同。

表三：推荐性能。每一行中表现最好的方法是黑体字，每一行中第二好的方法是下划线。对非神经方法和神经方法的改进分别显示在最后两栏。

数据集	衡量标准	(a) 录音带	(b) BPR	(c) FMC	(d) FPMC	(e) 基金会	(f) GRU4Rec	(g) GRU4Rec+	(h) 卡塞尔	(i) SASRec	改善对。 (a)-(e)	(f)-(h)
美的	撞击@10	0.4003	0.3775	0.3771	0.4310	0.4607	0.2125	0.3949	0.4264	**0.4854**	5.4%	13.8%
	NDCG@10	0.2277	0.2183	0.2477	0.2891	0.3020	0.1203	0.2556	0.2547	**0.3219**	6.6%	25.9%
游戏	击球@10	0.4724	0.4853	0.6358	0.6802	0.6838	0.2938	0.6599	0.5282	**0.7410**	8.5%	12.3%
	NDCG@10	0.2779	0.2875	0.4456	0.4680	0.4557	0.1837	0.4759	0.3214	**0.5360**	14.5%	12.6%
蒸汽	撞击@10	0.7172	0.7061	0.7731	0.7710	0.7624	0.4190	0.8018	0.7874	**0.8729**	13.2%	8.9%
	NDCG@10	0.4535	0.4436	0.5193	0.5011	0.4852	0.2691	0.5595	0.5381	**0.6306**	21.4%	12.7%
ML-1M	命中率@10	0.4329	0.5781	0.6986	0.7599	0.6413	0.5581	0.7501	0.7886	**0.8245**	8.5%	4.6%
	NDCG@10	0.2377	0.3287	0.4676	0.5176	0.3969	0.3381	0.5513	0.5538	**0.5905**	14.1%	6.6%

为了公平比较，我们使用*TemsorFlow*和Adam[41]优化器来实现BPR、FMC、FPMC和TransRec。对于GRU4Rec，GRU4Rec+，和Caser，我们使用相应作者提供的代码。对于除PopRec以外的所有方法，我们考虑了10、20、30、40、50的潜在*维度*。对于BPR、FMC、FPMC和TransRec，ℓ_2正则器从0.0001、0.001、0.01、0.1*和*1中选择。所有其他的超参数和初始化策略都是这些方法的作者所建议的。我们使用验证集来调整超参数，如果验证性能在20个epochs内没有改善，就终止训练。

C. 实施细节

对于SASRec默认版本的架构，我们使用两个自我注意块（$b = 2$），并使用学习的位置嵌入。嵌入层和预测层的项目嵌入是共享的。我们用*TensorFlow*实现SASRec。优化器是*Adam*优化器[41]，学习率设置为0.001，批次大小为128。由于*MovieLens-1m*的稀疏性，关闭神经元的辍学率为0.2，其他三个数据集为0.5。对于*MovieLens-1m*，最大序列长度n被设定为200，对于其他三个数据集，最大序列长度n被设定为50，即大致与每个用户的平均动作数成正比。下面将考察变体和不同超参数的性能。

所有代码和数据都应在出版时发布。

D. 评价指标

我们采用两个常见的Top-N指标，Hit Rate@10和NDCG@10，来评估推荐性能[14]，[19]。Hit@10计算的是真实的下一个项目在前10个项目中的次数，而NDCG@10是一个位置感知指标，对较高的位置分配较大的权重。请注意，由于我们对每个用户只有一个测试项目，Hit@10等同于Recall@10，并与Precision@10成正比。

为了避免对所有用户-项目对进行繁重的计算，我们采用了[14]，[48]中的策略。对于每个用户u，我们随机抽取100个负面项目，并将这些项目与基础事实项目进行排名。基于这101个项目的排名，Hit@10和NDCG@10可以被评估。

E. 建议 业绩

表三显示了所有方法在四个数据集上的推荐性能（**RQ1**）。通过考虑所有数据集中第二好的方法，出现了一个普遍的模式，非神经方法（即（a）-（e））在稀疏的数据集上表现更好，而神经方法（即（f）-（h））在密集的数据集上表现更好。据推测，这是因为神经方法有更多的参数来捕捉高阶转换（也就是说，它们表现力强，但容易过度拟合），而精心设计但更简单的模型在高稀疏度环境中更有效。

我们的方法SASRec在稀疏和密集的数据集上都优于所有基线，与最强的基线相比，获得了6.9%的命中率和9.6%的NDCG改进（平均）。一个可能的原因是，我们的模型可以在不同的数据集上适应性地参加不同范围的项目（例如，在稀疏数据集上只参加前一个项目，在密集数据集上参加更多的项目）。我们在第四节H中进一步分析这一行为。

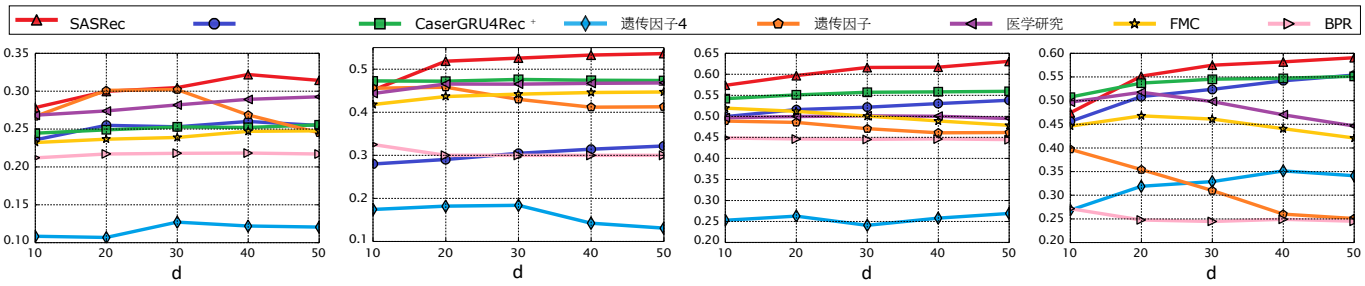
在图2中，我们还分析了一个关键的超参数--潜伏维数d的影响，显示了所有方法在d从10到50变化时的NDCG@10。我们看到，我们的模型通常受益于较大的潜伏维数。对于所有的数据集，我们的模型都取得了令人满意的成绩。

$d \geq 40$的性能。

F. 消融研究

由于我们的架构中有许多组件，我们通过消融研究来分析它们的影响（**RQ2**）。表四显示了我们的默认方法及其**8个**变体在所有四个数据集（d=50）上的表现。我们分别介绍了这些变体并分析了它们的效果。

- (1) *删除PE（位置嵌入）*。如果没有位置嵌入**P**，每个项目的关注权重只取决于项目嵌入。也就是说，模型根据用户过去的行为进行推荐，但其顺序并不重要。这个变体可能适用于稀疏的数据集，其中用户序列通常很短。这个变体在最稀疏的数据集（*Beauty*）上的表现比默认模型好，但在其他密集的数据集上表现较差。



(a) *靓丽* (b) *游戏* (c) *蒸汽* (d) *ML-1M*

图2：潜在维度d对排名性能的影响（NDCG@10）。

表四：四个数据集的消融分析（NDCG@10）。优于默认版本的性能以黑体字标出。"表示性能严重下降（超过10%）。

建筑学	*靓丽*	*游戏*	*蒸汽*	*ML-1M*
(0) 默认	0.3142	0.5360	0.6306	0.5905
(1) 移除PE	**0.3183**	0.5301	0.6036	0.5772
(2) 不共享的IE	0.*2437*↓	0.*4266*↓	0.*4472*↓	0.*4557*↓
(3) 移除RC	0.*2591*↓	0.*4303*↓	0.5693	0.5535
(4) 移除辍学	0.*2436*↓	0.*4375*↓	0.5959	0.5801
(5) 0块 (b=0)	0.*2620*↓	0.*4745*↓	0.*5588*↓	0.*4830*↓
(6) 1块(b=1)	0.3066	**0.5408**	0.6202	0.5653
(7) 3个区块 (b=3)	0.3078	0.5312	0.6275	**0.5931**
(8)多头型	0.3080	0.5311	0.6272	0.5885

- (2) *不共享的IE（项目嵌入）*。我们发现，使用两个项目嵌入会持续损害性能，可能是由于过度拟合。
- (3) *删除RC（残余连接）*。没有残余连接，我们发现性能明显变差。据推测，这是因为较低层的信息（如最后一个项目的嵌入和第一个区块的输出）不容易传播到最后一层，而这些信息对做推荐非常有用，特别是在稀疏的数据集上。
- (4) *移除Dropout*。我们的结果表明，dropout可以有效地规范化模型，以实现更好的测试性能，特别是在稀疏的数据集上。这些结果也意味着在密集数据集上的过拟合问题不那么严重。
- (5)-(7) *区块的数量*。不出所料，零块的结果较差，因为模型只取决于最后一个项目。一个区块的变体表现得相当好，尽管使用两个区块（默认模型）仍能提高性能，特别是在密集的数据集上，这意味着分层的自我注意结构有助于学习更复杂的项目转换。使用三个区块可以达到与默认模型类似的性能。
- (8) *多头*。Transformer[3]的作者发现，使用"多头"注意是有用的，它在h个子空间（每个子空间为d/h维空间）中应用注意。然而，使用两个头的性能是一致的，而且在我们的情况下，比单头注意力稍差。这可能要归功于我们问题中的小d（Transformer中d=512），它不适合分解成更小的子空间。

G. 培训效率和可扩展性

我们评估了我们模型的训练效率（**RQ3**）的两个方面。训练速度（一个历时的训练所需时间）和收敛时间（达到满意的性能所需时间）。我们还考察了我们的模型在最大长度n方面的可扩展性。所有的实验都是在单个GTX-1080 Ti GPU上进行的。

训练效率。图3展示了基于深度学习的方法在GPU加速下的效率。由于GRU4Rec的性能较差，所以省略了。为了公平比较，Caser和GRU4Rec有两个训练选项+：使用完整的训练数据或只使用最近的200个动作（如SASRec）。就计算速度而言，SASRec在一个epoch的模型更新上只花了1.7秒，比Caser（19.1s/epoch）快11倍多，比GRU4Rec+（30.7s/epoch）快18倍。我们还看到，SASRec在*ML-1M*上约350秒内收敛到最佳性能，而其他模型需要更长的时间。我们还发现，使用完整数据会使Caser和GRU4Rec+的性能更好。

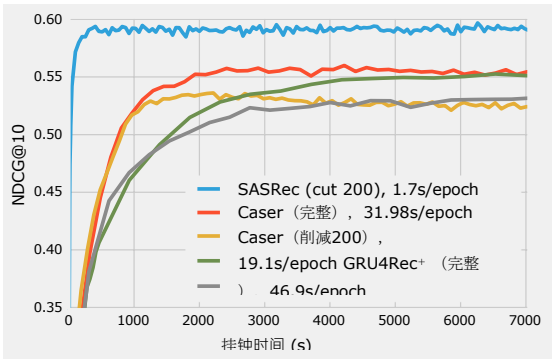
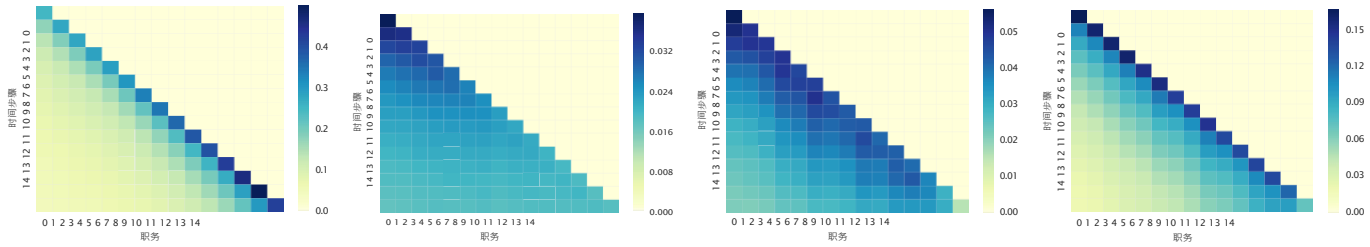


图3：*ML-1M*的训练效率。从每个历时和总的训练时间来看，SASRec比基于CNN/RNN的推荐方法快一个数量级。



(a) *美容*，第1层　(b) *ML-1M*，第1层，不含PE　(c) *ML-1M*，第1层　(d) *ML-1M*，第2层

图4：不同时间步骤的位置上的平均注意力权重的可视化。为了比较，基于一阶马尔科夫链的模型的热图将是一个对角线矩阵。

可扩展性。与标准的MF方法一样，SASRec与用户、项目和行动的总数量呈线性扩展。一个潜在的可扩展性问题是最大长度n，然而计算可以有效地与GPU并行。在这里，我们测量了不同n的SASRec的训练时间和性能，实证研究其可扩展性，并分析它是否能在大多数情况下处理顺序推荐。表五显示了不同序列长度的SASRec的性能和效率。n越大，性能越好，直到n=500左右，此时性能达到饱和（可能是因为99.8%的行动已经被覆盖）。然而，即使n=600，该模型也能在2000秒内完成训练，这仍然比Caser和GRU4Rec快[+]。因此，我们的模型可以很容易地扩展到多达几百个动作的用户序列，这适合于典型的评论和购买数据集。我们计划在未来研究处理非常长的序列的方法（在III-F节讨论）。

表五:可扩展性：*ML-1M*上不同最大长度n的性能和训练时间。

n	10	50	100	200	300	400	500	600
时间(s)	75	101	157	341	613	965	1406	1895
NDCG@10	0.480	0.557	0.571	0.587	0.593	0.594	0.596	0.595

H. 视觉化的注意权重

回顾一下，在时间步骤t，我们模型中的自我注意机制根据位置嵌入和项目嵌入对前t个项目进行适应性分配权重。为了回答**RQ4**，我们检查了所有的训练序列，并试图通过显示位置和项目的平均注意力权重来揭示有意义的模式。

对位置的关注。图4显示了过去15个时间步骤中最后15个位置上的平均注意力权重的四张热图。请注意，当我们计算平均权重时，分母是*有效*权重的数量，这样可以避免短序列中填充项的影响。

我们考虑在热图之间进行一些比较。

- *(a)与(c)*。这一比较表明，该模型倾向于在稀疏数据集*Beauty*上关注更多最近的项目，而在密集数据集*ML-1M*上*关注*更少的项目。这是我们的模型能够自适应地处理稀疏和密集数据集的关键因素，而现有的方法往往集中在光谱的一端。
- *(b)与(c)*。这个比较显示了使用位置嵌入（PE）的效果。没有它们，注意力权重基本上是均匀地分布在以前的项目上，而默认模型（c）在位置上更敏感，因为它倾向于关注最近的项目。
- *(c)与(d)*。由于我们的模型是分层的，这显示了注意力在不同区块之间的变化。显然，高层次的注意力倾向于集中在较近的位置。据推测，这是因为第一个自我注意区块已经考虑了所有以前的项目，而第二个区块不需要考虑遥远的位置。

总的来说，可视化显示，我们的自我注意机制的行为是*自适应的*，*有位置意识的*，并且是*分层次的*。

项目之间的注意。 显示注意力的权重为了进行更广泛的比较，我们使用了MovieLens-1M，其中每部电影都有几个类别，我们随机选择了两个互不相干的集合，每个集合包含4个类别的200部电影。为了进行更广泛的比较，使用*MovieLens-1M*，每部电影都有几个类别，我们随机选择两个不相干的集合，每个集合包含4个类别的200部电影。*科幻小说（Sci-Fi）*、*浪漫主义*、*动画*和*恐怖*。第一个集合用于查询，第二个集合作为关键。图5显示了两组之间平均注意力权重的热图。我们可以看到热图近似于一个块状对角线矩阵，这意味着注意力机制可以识别类似的项目（例如共享一个共同类别的项目），并倾向于在它们之间分配更大的权重（事先不知道类别）。

V. 总结

在这项工作中，我们提出了一个新颖的基于自我注意的序列模型*SASRec*，用于下一个项目的推荐。SASRec对整个用户序列进行建模（没有任何递归或卷积操作），并自适应地考虑消耗的项目进行预测。在稀疏和密集数据集上的大量经验结果表明，我们的模型优于先进的基线，并且比基于CNN/RNN的方法快一个数量级。在未来，我们计划通过纳入丰富的上下文信息（如停留时间、行动类型、地点、设备等）来扩展该模型，并研究处理很长的序列（如点击）的方法。

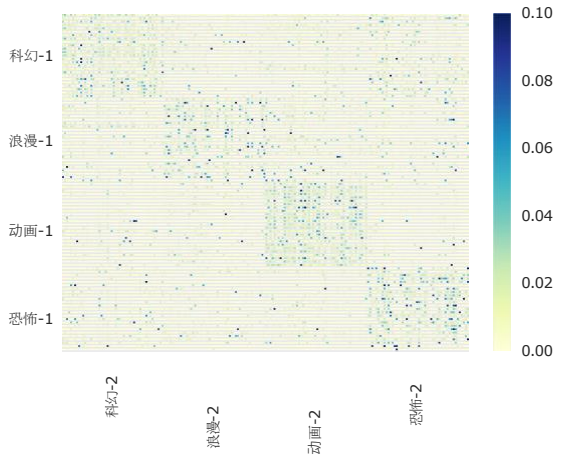


图5：四个类别的电影之间的平均注意力的可视化。这表明我们的模型可以发现项目的属性，并在相似的项目之间分配更大的权重。

参考文献

[1] S.Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized markov chains for next-basket recommendation, " in *WWW*, 2010.

[2] B.Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks, " in *ICLR*, 2016.

[3] A.Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L.Kaiser, and I. Polosukhin, "Attention is all you need," in *NIPS*, 2017.

[4] Y.Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets, " in *ICDM*, 2008.

[5] S.Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: bayesian personalized ranking from implicit feedback, " in *UAI*, 2009.

[6] F.Ricci, L. Rokach, B. Shapira, and P. Kantor, *Recommender systems handbook*.Springer US, 2011.

[7] Y.Koren和R.Bell，"协作过滤的进展"，载于*Recom- mender Systems Handbook*。Springer, 2011.

[8] S.Kabbur, X. Ning, and G. Karypis, "Fism: Factored item similarity models for top-n recommender systems, " in *SIGKDD*, 2013.

[9] S.Zhang, L. Yao, and A. Sun, "Deep learning based recommender system:A survey and new perspectives," *arXiv*, vol. abs/1707.07435, 2017.

[10] S.Wang, Y. Wang, J. Tang, K. Shu, S. Ranganath, and H. Liu, "What your images reveal:利用视觉内容进行兴趣点推荐，"在*WWW*，2017。

[11] W.Kang, C. Fang, Z. Wang, and J. McAuley, "Visually-aware fashion recommendation and design with generative image models, " in *ICDM*, 2017.

[12] H.Wang, N. Wang, and D. Yeung, "Collaborative deep learning for recommender systems, " in *SIGKDD*, 2015.

[13] D.H. Kim, C. Park, J. Oh, S. Lee, and H. Yu, "Convolutional matrix factorization for document context-aware recommendation, " in *RecSys*, 2016.

[14] X.He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. Chua, "Neural collaborative filtering," in *WWW*, 2017.

[15] S.Sedhain, A. K. Menon, S. Sanner, and L. Xie, "Autorec:Autoencoders满足协同过滤，"在*WWW*，2015。

[16] Y.Koren, "具有时间动态的协同过滤", *Communica- tions of the ACM*, 2010.

[17] C.Wu, A. Ahmed, A. Beutel, A. J. Smola, and H. Jing, "Recurrent recommender networks," in *WSDM*, 2017.

[18] L.L. Xiong, X. Chen, T. -K.Huang, J. Schneider, and J. G. Carbonell, "Temporal collaborative filtering with bayesian probabilistic tensor factorization, " in *SDM*, 2010.

[19] R.He, W. Kang, and J. McAuley, "基于翻译的推荐，" in *RecSys*, 2017.

[20] R.He, C. Fang, Z. Wang, and J. McAuley, "Vista:视觉上、社会上和时间上感知的艺术推荐模型，"在*RecSys*，2016。

[21] R.He and J. McAuley, "Fusing similarity models with markov chains for sparse sequential recommendation," in *ICDM*, 2016.

[22] J.Tang和K. Wang, "Personalized top-n sequential recommendation via convolutional sequence embedding," in *WSDM*, 2018.

[23] H.Jing和A. J. Smola，"神经生存推荐器"，在*WSDM*，2017。

[24] Q.Liu, S. Wu, D. Wang, Z. Li, and L. Wang, "Context-aware sequential recommendation, " in *ICDM*, 2016.

[25] A.Beutel, P. Covington, S. Jain, C. Xu, J. Li, V. Gatto, and E. H. Chi, "latent cross:Making use of context in recurrent recommender systems," in *WSDM*, 2018.

[26] B.Hidasi和A. Karatzoglou, "Recurrent neural networks with top-k gains for session-based recommendations," *CoRR*, vol. abs/1706.03847, 2017.

[27] K.Xu, J. Ba, R. Kiros, K. Cho, A. C. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio, "Show, attend and tell:用视觉注意力生成神经图像标题，"在*ICML*，2015。

[28] D.Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *ICLR*, 2015.

[29] J.Chen, H. Zhang, X. He, L. Nie, W. Liu, and T. Chua, "Attentive collaborative filtering:多媒体推荐与项目和组件级的关注，"在*SIGIR*，2017。

[30] J.Xiao, H. Ye, X. He, H. Zhang, F. Wu, and T. Chua, "注意力分解机。通过注意力网络学习特征交互的权重，"在*IJCAI*，2017。

[31] S.Wang, L. Hu, L. Cao, X. Huang, D. Lian, and W. Liu, "Attention-based transactional context embedding for next-item recommendation, " in *AAAI*, 2018.

[32] Y.Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M.Krikun, Y. Cao, Q. Gao, K. Macherey等人，"谷歌的神经机器翻译系统。弥合人类和机器翻译之间的差距，"*arXiv预印本arXiv:1609.08144*，2016。

[33] J.Zhou, Y. Cao, X. Wang, P. Li, and W. Xu, "Deep recurrent models with fast-forward connections for neural machine translation, " *TACL*, 2016.

[34] M.D. Zeiler和R. Fergus，"可视化和理解卷积网络"，在*ECCV*，2014。

[35] K.He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[36] L.J. Ba, R. Kiros, and G. E. Hinton, "Layer normalization," *CoRR*, vol. abs/1607.06450, 2016.

[37] S.Ioffe and C. Szegedy, "Batch normalization:通过减少内部协变量偏移加速深度网络训练，"在*ICML*，2015。

[38] N.Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhut- dinov, "Dropout: a simple way to prevent neural networks from overfitting, " *JMLR*, 2014.

[39] D.Warde-Farley, I. J. Goodfellow, A. C. Courville, and Y. Bengio, "An empirical analysis of dropout in piecewise linear networks," *CoRR*, vol. abs/1312.6197, 2013.

[40] Y.Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, 2009.

[41] D.P. Kingma和J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

[42] D.Povey, H. Hadian, P. Ghahremani, K. Li, and S. Khudanpur, "A time-restricted self-attention layer for asr, " in *ICASSP*, 2018.

[43] P.Wang, J. Guo, Y. Lan, J. Xu, S. Wan, and X. Cheng, "Learning hierarchical representation model for next basket recommendation, " in *SIGIR*, 2015.

[44] S.Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *CoRR*, vol. abs/1803.01271, 2018.

[45] S.Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber *et al.*, "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," 2001.

[46] J.J. McAuley, C. Targett, Q. Shi, and A. van den Hengel, "基于图像的风格和替代品推荐，" 在*SIGIR*, 2015。

[47] S.Feng, X. Li, Y. Zeng, G. Cong, Y. M. Chee, and Q. Yuan, "Personalized ranking metric embedding for next new poi recommendation, " in *IJCAI*, 2015.

[48] Y.Koren, "因式分解与邻里关系：一个多层面的协作过滤模型"，在*SIGKDD*, 2008.