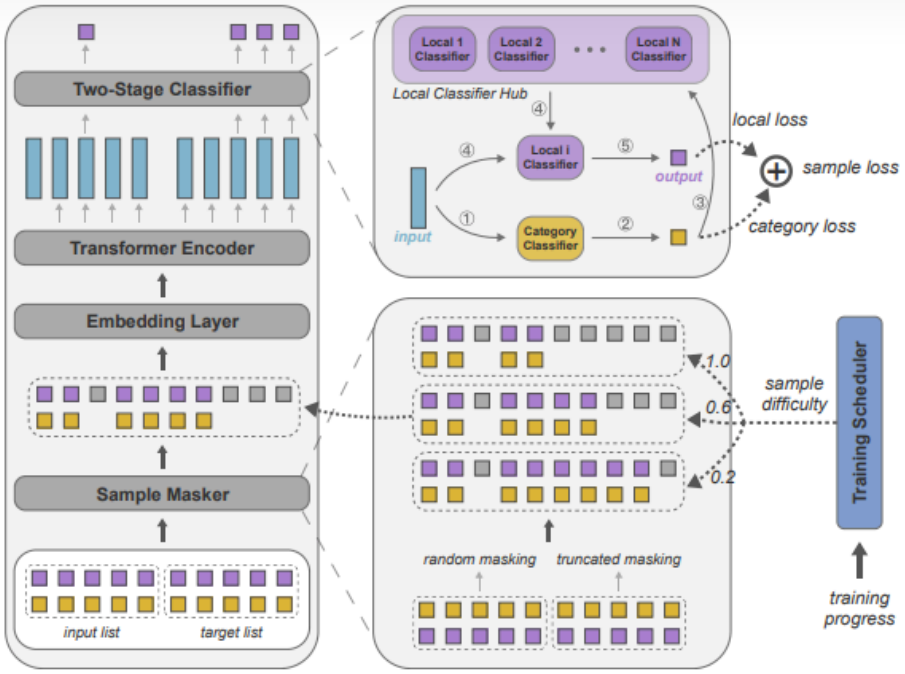
知乎



华为-2023：FANS-解锁自回归的力量：揭秘如何利用自回归完成序列推荐任务

SmartMindAI
专注搜索、广告、推荐、大模型和人工智能最新技术，欢迎关注我

已关注

2 人赞同了该文章

Introduction

在许多在线内容平台中，用户自建的列表已经取代了官方列表作为系统性的内容分享的一部分。这些列表中的项目可以根据用户的偏好、类型、创作者等进行组织。例如，在知乎上，用户将机器学习教程中的多个答案组成一个主题（即列表），而在Spotify上，用户会将Rihanna的著名歌曲组成一个播放列表。 为了帮助"列表构建者"创建长列表以推荐潜在的后续列表，或者继续提供相关项列表给"列表消费者"完成当前列表后，项列表续订系统应运而生，使他们不必费力地在大量的项中搜索相关的内容。

挑战： 给定一个输入项列表，列表延续系统的目标是理解序列信息并生成一个与前一项相连续的项列表。这项任务的关键挑战有两个：一是如何基于长期趋势预测项而不是特定之前的项；二是如何在一个请求中返回多个项以减少工业系统的服务器开销，同时考虑到项之间的相互关系和顺序。

之前的工作： 虽然对序列推荐的研究已经被广泛研究，但在序列推荐（即用户行为的下一个项目预测）和列表延续（即维护用户创建的列表的一致性）之间存在差距。现有的序列推荐模型通常被用作召回器进行下一个项目的推荐，并返回最大概率的前K个候选项。因此，直接使用这些模型来生成列表延续是不合适的，因为它们无法捕捉项的顺序和它们之间的关系在生成的列表中。为了解决这个问题，一种直接的解决方案是采用自回归策略，如图 所示，逐个生成项列表。与自然语言处理+（NLP）中的语言建模任务类似，它将全局分类中的最大概率项作为预测。现有的列表延续方法也是基于这种策略设计的，但不幸的是由于需要高效率部署到生产环境，这可能会导致模型推理时间的大量增加，且随着生成序列长度的增加和项词汇量的增大呈线性和二次方级数增加。由于推荐系统+的项词汇量远大于NLP中的词词汇量，这是由于项的不可分性，softmax分类器将遭受严重的计算复杂度问题。

知乎

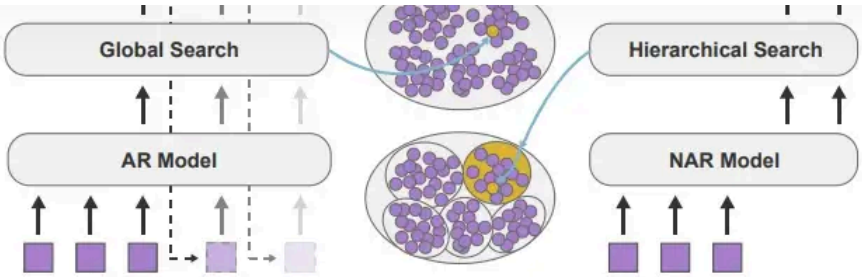


Figure 1: Differences between previous autoregressive (AR) models with global search and our proposed non-autoregressive (NAR) model with hierarchical search.

当前工作： 在这篇论文中，我们采用了非自回归的方法来加速推理效率和提高推理质量。如图所示，我们通过非自回归的生成策略同时解码接下来的K项以及层次搜索方法进行模糊匹配和精确定位来解决以前模型的不足。提出了一种名为 FANS 的模型，它是基于双向Transformer的快速非自回归序列生成模型，区别于传统的自回归BERT4Rec模型，它一次性插入多个掩码token并进行一次过程。同时，我们设计了一个两阶段的分类器来层次地解码隐含表示。此外，我们还提议使用逐步学习的方式来训练我们的模型，以提高非自回归生成的推理质量。这些贡献将在方法部分进行详细介绍。

- 我们首次利用非自回归生成方法处理项目列表的延续，旨在提高推理效率并满足工业系统的需求。
- 我们提出了一个名为FANS的新模型，它采用非自回归生成方法和两阶段分类器进行层次搜索，以提高推理效率，并利用基于课程学习的训练调度器提升推理质量。
- 本文评估了FANS模型在四个真实数据集上的表现，并通过在线工业系统的AB测试验证其效果。实验结果表明，FANS在大部分指标上表现最优，且能带来高达8.7倍的效率提升。

Proposed Method

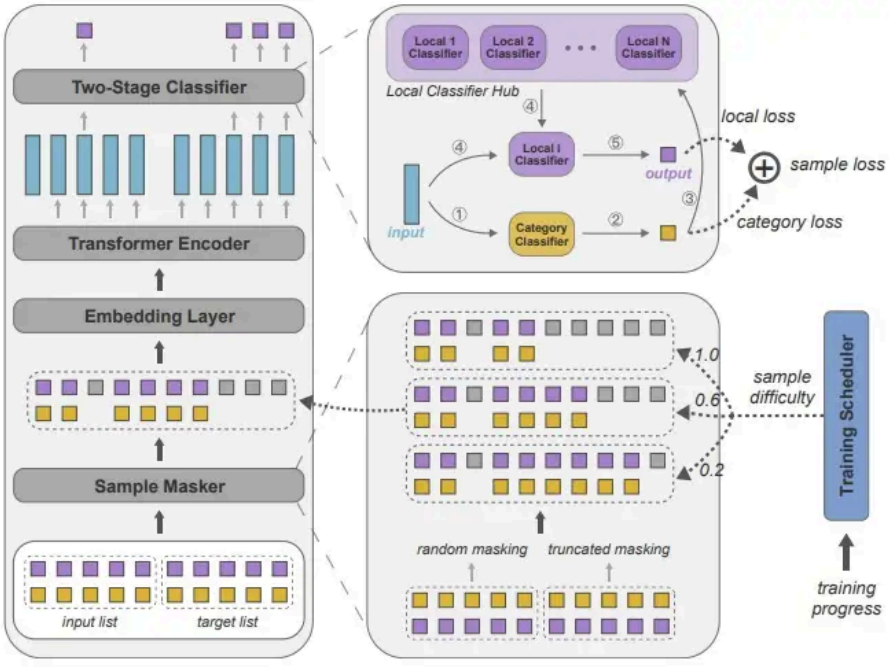


Figure 2: Framework of our FANS model. Purple and yellow squares represent item and category tokens respectively, grey squares represent masked tokens, and blue bars indicate latent embeddings. The category tokens of masked items are set to ⟨PAD⟩ tokens, which are not shown.

知乎

在序列推荐中，顺序项序列建模被广泛应用并取得成功。Transformer-based模型（如SASRec、CAR和BERT4Rec）已成功用于序列推荐系统。然而，这些模型通常被视为召回器，以获取最高的前-K个候选项目。相反，一种直接方法是将这些模型视为生成器⁺并进行自回归式的序列生成。这种方法可以很好地保留原模型特性，通过调整预测步长控制生成文本长度。然而，这种方法也存在一些问题，如预测结果影响下一节点选择可能导致不连贯或不合理结果，而且可能受训练数据不足影响，因为训练数据通常包含少量的序列数据点。

$$p\left(\mathbf{y}'=\mathbf{y}|\mathbf{x}\right)=\prod_{i=1}^{|\mathbf{y}|}p\left(y_i'=y_i|\mathbf{x},\mathbf{y}_{1:i-1}\right).$$

然而，自回归方法需要逐个解码，限制了其在工业场景中的应用，因为推理效率低。为了解决这个问题，本文提出了一种非自回归方法FANS，用于并行序列生成，并提高推理效率。

$$p\left(\mathbf{y}'=\mathbf{y}|\mathbf{x}\right)=\prod_{i=1}^{|\mathbf{y}|}p\left(y_i'=y_i|\mathbf{x}\right).$$

图显示了我们的模型结构，包含4个基本模块：Sample Masker、Embedding Layer、Transformer Encoder和Two-phase Classifier。特别地，FANS接受目标列表对和类别的输入。Sample Masker首先处理输入的目标对，遮蔽某些部分项，生成不同难度级别的token级样本。接着，Embedding Layer将token序列⁺转换为密集的嵌入序列。然后，Transformer Encoder模型嵌入这个序列以学习上下文中的项目表示。最后，Two-phase Classifier以层次化的形式解码输出嵌入并计算样本损失。 此外，为了提高非自回归生成的推理质量，我们采用基于课程的学习策略进行优化训练。实际上，这是一款训练调度器，可以控制Sample Masker的难度级别。

Non-autoregressive Generation

使用Transformer模型对连续子序列进行模型，并应用Cloze任务对目标列表进行掩蔽。使用混合项目遮蔽策略进行训练，包括对输入列表和目标列表进行随机和截断遮蔽。在每个样本中，都有一个项目列表和一个类别列表。对项目列表进行遮蔽时，会进行三种遮蔽操作之一：替换为特殊标记、从词汇表⁺中随机选择项目或保持不变。对目标列表进行遮蔽时，将最后的一定比例的项目替换为特殊标记。在对类别列表进行调整时，如果将某个项目的类别的类令牌进行遮蔽，那么对应的类令牌会被替换为特殊标记。通过定义ρ_t来衡量样本难度。ρ_t越接近0，样本越容易；ρ_t越接近1，样本越困难。在第一个步骤中，将创建四种类型的token序列：item, category, position和segment token序列。之后，将这些token序列从one-hot向量转换为统一长度的低维密集向量。最后，将这四个token嵌入合并为一个输入嵌入供Transformer编码器使用。

$$\boldsymbol{u}=[x_1,\cdots,\langle\mathbf{MASK}\rangle,\cdots,x_{|\mathbf{x}|},y_1,\cdots,\langle\mathbf{MASK}\rangle,\cdots\langle\mathbf{MASK}\rangle].$$

同样地，可以构造出的类别标记序列如下：

$$\hat{\mathbf{z}}^t=\mathbf{w}^T\hat{\mathbf{x}}^t$$

其中$\hat{\mathbf{x}}^t=[x_1^t,x_2^t,\ldots,x_n^t]$表示第t时刻输入序列中的每一个特征值。

$$\boldsymbol{c}=[c_{x_1},\cdots,\langle\mathbf{PAD}\rangle,\cdots,c_{x_{|\mathbf{x}|}},c_{y_1},\cdots,\langle\mathbf{PAD}\rangle,\cdots\langle\mathbf{PAD}\rangle].$$
$$\boldsymbol{E}^0=\boldsymbol{A}_u+\boldsymbol{A}_s+\boldsymbol{A}_p+\boldsymbol{A}_s\in\mathbb{R}^{l\times d}$$

Transformer Encoder: 我们使用Transformer网络作为基础结构。每个Transformer层⁺都有独立的权重，但结构保持一致。通过多层Transformer层编码后，得到输出序列$\boldsymbol{E}^H$。$\boldsymbol{A}_u$、$\boldsymbol{A}_c$、$\boldsymbol{A}_p$和$\boldsymbol{A}_s$是经过转换后的项目、类别、位置和段落的token嵌入。

$$\boldsymbol{E}^h=\mathrm{TRANS}^h\left(\boldsymbol{E}^{h-1}\right),h=1,2,\ldots,H,$$

H表示Transformer层数量，TRANS^h表示第h层Transformer。

Two-stage Classifier

已知MLM任务使用vanilla分类器，该分类器直接通过softmax函数对整个词典$\mathcal{V}$的每个元素进行概率分布估计。softmax函数定义为：$p(v_i|v_1,\cdots,v_{i-1},\boldsymbol{x})$，表示词汇索引为$v_i$的概率，前缀词语为$v_1,\cdots,v_{i-1}$，输入向量为$\boldsymbol{x}$。

知乎

在项目列表延续的情况下，针对词汇量大导致计算复杂性⁺和空间复杂性增加的问题，一种解决方案是使用基于内容相似性的两阶段分类器(TSC)。TSC模块由两个部分组成：类别分类器用于初步分类，本地分类器中心包含N个本地分类器用于细化搜索。该模型可以提高分类效率并降低搜索成本。

$$\mathrm{C}_{cat}:\mathbb{R}^d\to\mathbb{R}^N,$$

$$\mathrm{C}_{loc}^i:\mathbb{R}^d\to\mathbb{R}^{M^i}.$$

如图1所示，设t为样本，其中遮盖的token为u_k。给定潜在嵌入E_j^H，类别分类器预测u_k的类别c_{u_k}，损失函数⁺定义为：

$$L=-\sum_{j=1}^{M}c_{u_k}\log y_j,$$

其中$y_j=g(E_j^H)$是模型输出的概率，$g(\cdot)$是激活函数⁺，M是潜在嵌入的数量。

$$L_{\mathrm{cat}}=-\frac{1}{n}\sum_{i=1}^{n}\sum_{z\in t}\log\left(\mathrm{C}_{cat}\left(E_j^H\right)_{c_{u_k}}\right),$$

对于每个类别的所有被遮盖项，选择类别中获得最大概率的本地分类器，并将其潜在嵌入投影到类别空间。局部分类器的损失函数如下：

$$L(c_{u_k},z)=-\log p(c_{u_k}|z)=-\log\frac{\exp(f(z|c_{u_k})^Tw_{c_{u_k}})}{\sum_{c_l}\exp(f(z|c_l)^Tw_{c_l})}$$

$$\Phi(z)=1-\left|\mathrm{sgn}\left(\arg\max_{i\in(1,\ldots,N)}\mathrm{C}_{cat}\left(E_j^H\right)_i-c_{u_k}\right)\right|,$$

$$L_{\mathrm{loc}}^{c_{u_k}}=-\frac{\sum_{i=1}^{n}\sum_{z\in t}\Phi(z)\log\left(\mathrm{C}_{loc}^{c_{u_k}}\left(E_j^H\right)_{u_k}\right)}{\sum_{i=1}^{n}\sum_{z\in t}\Phi(z)+\epsilon}.$$

：$\Phi(z)$为布尔检测器，检查分类器答案正确性；sgn为符号函数，防止分母为0；损失函数可表示为：

$$J(w)=-\frac{1}{m}\sum_{i=1}^{m}\left[\mathrm{sgn}\left(h(x_i)\cdot y_i\right)-y_i\right]^2+\frac{\lambda}{2m}\|w\|^2$$

$$L=L_{\mathrm{cat}}+\frac{1}{N}\sum_{i=1}^{N}L_{\mathrm{loc}}^i.$$

Curriculum Learning Based Training

图展示了非自回归生成的训练策略和我们提出的逐步训练策略，每个策略对应一个训练调度器。

naive调度器简单地让样本遮罩器在所有训练阶段生成大量的硬样本。因此，样本难度ρ_t始终为1。

step-wise调度器选择了一种从容易到困难的学习方法，要求样本遮罩器在不同的训练阶段生成一定难度水平的样本。调度器首先将所有的训练阶段划分为相等长度的部分，每个部分（包含一定的训练阶段）被视为*一种课程步骤*。在每个部分中，样本难度ρ_t是恒定的。随着训练的进行，ρ_t逐渐增加，即更难的样本将用于训练FANS模型。

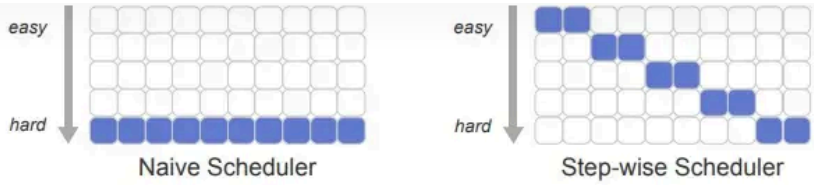


Figure 3: Naive scheduler and proposed step-wise scheduler. Each square represents the sample weight at a different difficulty level, which is used for loss calculation during training. Precisely, at each training epoch, naive scheduler converts all training samples to hard samples, while step-wise scheduler converts all training samples to some difficulty level that changes gradually from easy to hard as training goes on.

Experiments

Experimental Setup

删除频率小于10的项，截取或过滤最大和最小长度的数据集，将合格的列表均匀分两段（输入和目标），建立项目-类别映射，按照8:1:1比例划分训练集、验证集和测试集[+]。

Table 1: Dataset statistics.

	Zhihu	Spotify	AotM	Goodreads
#Lists	18,704	72,152	12,940	15,426
#Items	36,005	104,695	6,264	47,877
#Categories	10	10	10	10
#Interactions	927,781	6,809,820	162,106	1,589,480
Avg. # items per list	49.59	94.38	12.53	103.04
Range # items per list	10 ~ 200	20 ~ 300	10 ~ 60	20 ~ 300
Density	0.138%	0.089%	0.208%	0.215%

我们对比了四种模型：Caser（卷积神经网络[+]）、GRU4Rec（门控循环单元）、SASRec（Transformer）、BERT4Rec（Transformer）。Caser和GRU4Rec对序列进行建模；SASRec利用Transformer，而BERT4Rec则使用Transformer为主要框架。在训练过程中，BERT4Rec使用MLM任务理解上下文信息，并在推理时通过⟨MASK⟩标记预测。为了更好地模型项目一致性的特征，CAR（基于SASRec的Transformer基于项目的列表延续模型）提出了一种门控网络。由于数据集中只有匿名化用户生成列表，我们放弃了使用GUPM（通用用户偏好模型）网络，该网络使用用户ID特性能识别人。

Comparison with State-of-the-art Methods

比较了FANS模型与最先进的方法在四个项目列表延续数据集上的性能。表格总结了所有模型的总体性能。 速度提升 度量是将当前模型的延迟（ms）与BERT4Rec模型进行比较的结果。结果表明Transformer-based BERT4Rec模型在四个指标上（NDCG\@5、10和HR\@5、10）表现最好，而自回归模型[+]（蓝色）比非自回归机制加速的模型（橙色）效率低得多，且更无效。虽然，但其延迟等于。然而，由于基于步骤式的课程学习训练策略，它超过了$\text{BERT4Rec}_{\text{NAR}}$，并在16个案例中的13个案例中实现了最先进的性能。与BERT4Rec相比，（从2.4倍到3.0倍），但在大多数情况下表现优秀。这可能是因为自动回归生成的结果很容易受到之前生成的项的影响，导致序列偏离正常趋势；在非自回归方法的帮助下，对物品生成的长期趋势预测的准确性更高。然而，当项词典大小（例如Spotify数据集的104K项）过大时，（即延迟小于50ms）。 对于小数据集，粉丝V可以

知乎

Table 2: Comparison of our FANS model with state-of-the-art methods. Blue circle (●) indicates an autoregressive model. Orange circle (●) indicates using non-autoregressive generation. Green circle (●) indicates using both non-autoregressive generation and the proposed two-stage classifier. The "Speedup" metric is computed as the ratio of the latency (ms) of the present model and that of BERT4Rec. We bold the best result and underline the second best. Results with latency above 50ms are in red (i.e., fail to meet the efficiency requirement), and those below 50ms are in green (i.e., meet the efficiency requirement).

		● Caser	● GRU4Rec	● SASRec	● BERT4Rec	● CAR	● $\text{BERT4Rec}_{\text{NAR}}$ / $\text{FANS}_{\text{NAIVE}}$	● $\text{FANS}_{\text{STEP}}$	● $\text{FANS}_{\text{TSC-NAIVE}}$	● $\text{FANS}_{\text{TSC-STEP}}$
Zhihu	NDCG@5	0.0169	0.0077	0.0125	0.0226	0.0156	0.0216	**0.0256**	0.0192	0.0232
	NDCG@10	0.0212	0.0132	0.0194	0.0329	0.0228	0.0325	**0.0389**	0.0284	0.0337
	HR@5	0.1862	0.1049	0.1707	0.2477	0.1792	0.2399	**0.2857**	0.2295	0.2670
	HR@10	0.2960	0.2255	0.3081	0.4275	0.3248	0.4202	**0.4819**	0.3953	0.4604
	Latency (ms)	88.82	92.56	125.74	120.38	118.76	42.33	42.33	**18.39**	**18.39**
	Speedup	1.4x	1.3x	1.0x	1.0x	1.0x	2.8x	2.8x	**6.5x**	**6.5x**
Spotify	NDCG@5	0.0277	0.0067	0.0081	0.0211	0.0254	0.0272	0.0313	0.0034	**0.0315**
	NDCG@10	0.0393	0.0101	0.0113	0.0304	0.0366	0.0402	**0.0461**	0.0047	0.0438
	HR@5	0.3477	0.1293	0.1536	0.2811	0.3307	0.3867	**0.4071**	0.0711	0.3992
	HR@10	0.4857	0.2110	0.2286	0.4227	0.4755	0.5502	**0.5729**	0.1188	0.5552
	Latency (ms)	230.51	259.18	240.48	325.17	279.07	132.88	132.88	**37.46**	**37.46**
	Speedup	1.4x	1.3x	1.4x	1.0x	1.2x	2.4x	2.4x	**8.7x**	**8.7x**
AotM	NDCG@5	0.0112	0.0053	0.0104	0.0218	0.0072	0.0171	0.0214	**0.0229**	0.0220
	NDCG@10	0.0312	0.0127	0.0662	**0.1008**	0.0314	0.0890	0.0992	0.0744	0.0995
	HR@5	0.0635	0.0317	0.0565	0.1115	0.0433	0.0983	**0.1130**	0.1084	0.1053
	HR@10	0.2923	0.1538	0.4308	**0.5538**	0.2462	0.5385	**0.5538**	0.4769	0.4923
	Latency (ms)	10.84	10.84	17.03	18.58	17.80	7.70	7.70	**7.01**	**7.01**
	Speedup	1.7x	1.7x	1.1x	1.0x	1.0x	2.4x	2.4x	**2.7x**	**2.7x**
Goodreads	NDCG@5	0.0205	0.0074	0.0187	0.0134	0.0232	0.0257	**0.0334**	0.0276	0.0293
	NDCG@10	0.0278	0.0104	0.0271	0.0204	0.0332	0.0355	**0.0467**	0.0390	0.0418
	HR@5	0.2406	0.1154	0.2464	0.1958	0.2789	0.3104	**0.3891**	0.3171	0.3268
	HR@10	0.3294	0.1783	0.3632	0.3379	0.4196	0.4492	**0.5149**	0.4202	0.4514
	Latency (ms)	134.38	165.70	188.07	194.55	233.46	64.92	64.92	24.76	**24.76**
	Speedup	1.4x	1.2x	1.0x	1.0x	0.8x	3.0x	3.0x	**7.9x**	**7.9x**

Ablation Studies

Impact of Hyper-parameters on Inference Quality and Inference Efficiency

首先，我们研究了类别信息对FANS性能的影响。CAR*和BERT4Rec*分别表示在CAR和BERT4Rec中移除分类特征。由于两阶段分类器基于类别信息，我们使用vanilla分类器作为替代，以构造没有类别特征注入的模型，即FANS*。图1展示了三个模型在四个数据集上的推理质量影响，从其中我们可以得出以下观察结果。首先，通过比较CAR*和CAR，以及BERT4Rec*和BERT4Rec，可以看出在四个数据集上推理质量得到了提高，证明了类别信息的有效性。其次，在大多数情况下，FANS*优于fanSS模型，因为两阶段分类器缩小了项搜索空间，可能防止其他类别的正确项被选择。

Table 3: Influence of the training schedulers on inference quality. The experiments are conducted on the Zhihu dataset. "naive" means using the naive scheduler. "step=s" means using the step-wise scheduler with s curriculum steps.

classifier	scheduler	naive / step=1	step=3	step=5	step=7	step=10
Vanilla Classifier	NDCG@5	0.0216	0.0232	0.0256	0.0268	**0.0274**
	NDCG@10	0.0325	0.0344	0.0389	**0.0390**	0.0370
	HR@5	0.2399	0.2734	**0.2857**	0.2840	0.2820
	HR@10	0.4202	0.4671	**0.4819**	0.4792	0.4577
Two-stage Classifier	NDCG@5	0.0210	0.0226	**0.0232**	0.0224	0.0210
	NDCG@10	0.0312	0.0329	**0.0337**	0.0332	0.0310
	HR@5	0.2424	0.2627	**0.2670**	0.2611	0.2504
	HR@10	0.4101	0.4362	**0.4604**	0.4416	0.4201

1) BERT4Rec的推理时间不受类别词汇量影响；

2) FANS模型在类别大小大时比BERT4Rec模型快，证明非自回归模型效率更高；

知乎

4) 对于每个数据集，有不同阈值，当类别大小大于阈值时，类别分类器时间增加超过了局部分类器时间减少，阈值随item vocabulary大小增加而增加。理想情况下，Spotify的数据集可加速到11.9倍，Goodreads数据集可加速到9.2倍。

Assessment of Inference Efficiency in an Industrial System

我们验证了华为音乐列表续订系统的效率，该系统每天服务数百万用户。任务是在用户完成播放列表后提供后续项列表。由于长序列的项目和大量的音乐数据（过滤后的约240K），直接使用三层Bert4Rec模型将导致超过300ms的不可接受延迟（在Tesla P100 PCIe GPU上测试）。为了提高推理效率，我们应用我们的风扇S模型。我们使用类别分类器对音乐项进行分类。我们将类的数量设置为10（即粉丝S`[N=10]`）和100（即粉丝S`[N=100]`）并发现推理效率从300ms下降到42ms（7.1x）和28ms（10.7x），分别。与粉丝S`[N=10]`相比，粉丝S`[N=100]`的推理质量只低了1.2%，而推理速度是其1.5倍，这是完全可以在应用场景中接受的。

Conclusion

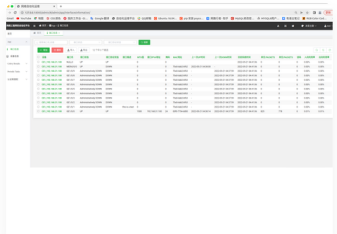
本文提出了一种FANS模型，通过非自反生成加速推理效率，并采用两阶段分类策略进行分层解码。该模型首次将非自反生成引入项列表续写，同时利用基于课程学习的训练策略提高非自反生成质量。实验证明，FANS模型在保持推理效率的同时，生成质量优于或等同于最先进的方法。

原文《FANS: Fast Non-Autoregressive Sequence Generation for Item List Continuation》

发布于 2024-01-10 16:52 · IP 属地北京

华为　序列推荐　非自回归生成研究

推荐阅读



基于手机信令的大数据分析教程（二）—— 数据库中职住数...

RedHat&CentOS救援模式/单用户模式