# FANS：面向物品列表续写的快速非自回归序列生成

Qijiong Liu
j
yonn.liu
@connect.polyu.hk 香港理工大学 中国香港特别行政区

Jieming Zhu
jiemingzhu@ieee.org 华为诺亚方舟实验室 中国深圳

Jiahao Wu
jiahao.wu@connect.polyu.hk
香港理工大学 中国香港特别行政区

Tiandeng Wu
wutiandeng1@huawei.com 华为技术有限公司 中国江苏

Zhenhua Dong
dongzhenhua
@huawei.com 华为诺亚方舟实验室 中国深圳

Xiao-Ming Wu*
xiao-ming.wu@polyu.edu.hk
香港理工
大学
中国香港特别行政区

arXiv:2304.00545v1 [cs.IR] 2 Apr 2023

## 摘要

用户整理的物品列表，例如 YouTube 上的视频播放列表和 Goodreads 上的图书列表，已成为在线平台上内容分享的常见形式。列表续写旨在对列表的整体趋势进行建模，并预测后续物品。近年来，基于 Transformer 的模型在理解上下文信息和捕捉列表中的物品关系方面展现出良好潜力。然而，将其部署到实时工业应用中仍具挑战，主要原因是它们采用的自回归生成机制耗时较长。本文提出一种新颖的快速非自回归序列生成模型 FANS，以提升物品列表续写的推理效率和生成质量。首先，我们采用非自回归生成机制，同时解码接下来的 $K$ 个物品，而非像现有模型那样逐个解码。随后，我们设计了一种两阶段分类器，取代当前基于 Transformer 的模型所用的基础分类器，进一步缩短解码时间。此外，为提高非自回归生成的质量，我们采用课程学习策略来优化训练。在知乎、Spotify、AotM 和 Goodreads 四个真实世界的物品列表续接数据集上的实验结果表明，与最先进的自回归模型相比，我们的 FANS 模型能够显著提升推理效率（最高达 8.7 倍），同时在物品列表续写方面实现具有竞争力或更优的生成质量。我们还在工业场景中验证了 FANS 的效率。我们的源代码和数据将发布于 MindSpore/models[1] 和 GitHub[2]。

## CCS 概念

• **信息系统** → **检索模型与排序**； •**计算方法** → **知识表示与推理**；多任务学习。

## 关键词

物品列表续写、非自回归生成、课程学习、序列推荐

**ACM 引用格式：**
Qijiong Liu、Jieming Zhu、Jiahao Wu、Tiandeng Wu、Zhenhua Dong 和 Xiao-Ming Wu。2023。FANS：面向物品列表续写的快速非自回归序列生成。载于《*2023*年*ACM Web*会议论文集（*WWW '23*）》，*2023*年*4*月*30*日至*5*月*4*日，美国德克萨斯州奥斯汀。ACM，美国纽约州纽约市，10 页。
https://doi.org/10.1145/3543507.3583430

## 1 引言

**物品列表续写的背景与应用**。 在许多在线内容平台中，用户策划列表已取代官方列表，成为系统化内容分享不可或缺的一部分。这些列表中的项目可根据用户偏好、类型、创作者等进行组织。例如，知乎[3] 用户会将机器学习教程中的多个回答归入一个专题（即列表），而 Spotify[4]用户则会将 Rihanna 的热门歌曲整理成播放列表。为了帮助列表创建者策划长列表，以推荐潜在的后续列表，或在列表使用者浏览完当前列表后继续提供相关物品列表，物品列表续接系统应运而生，使他们不必再从海量物品中费力搜索相关内容。

**挑战**。 给定一个输入物品列表，列表续接系统旨在理解列表中的序列信息，并生成一个延续前一列表的物品列表。这项任务的关键挑战有两个：1）如何根据序列中的长期趋势而非某个特定的前序物品来预测物品；2）如何在一次请求中返回多个物品，以降低工业系统中的服务器开销，同时考虑物品之间的相互关系及其顺序。

**已有研究**。 针对序列推荐中物品序列建模的研究已十分广泛 [13, 25, 26], 但

*Xiao-Ming Wu 是通讯作者。[1] https://gitee.com/mindspore/models/tree/master/research/recommend[2] https://github.com/Jyonn/FANS

[3]在线问答服务： https://www.zhihu.com[4]在线音乐流媒体服务： https://www.spotify.com

允许免费为个人或课堂使用而制作本作品的全部或部分数字副本或困难副本，前提是这些副本不得用于营利或商业利益，也不得为此目的进行复制或分发，并且副本首页须保留本声明和完整引文。对于本作品中由作者以外的其他方拥有版权的内容，必须尊重其版权。注明出处后可以摘录。以其他方式复制、再版、发布到服务器或分发至列表，均须事先获得具体许可和/或支付费用。请向 permissions@acm.org 申请许可。*WWW '23*，*2023*年*4*月*30*日至*5*月*4*日，美国德克萨斯州奥斯汀© 2023 版权所有，归所有者/作者所有。出版权许可给 ACM。
ACM ISBN 978-1-4503-9416-1/23/04...$15.00
https://doi.org/10.1145/3543507.3583430

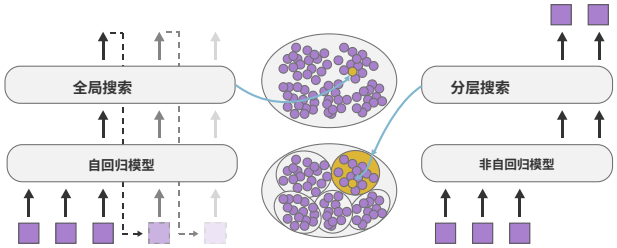



**图 1：采用全局搜索的先前自回归（AR）模型与我们提出的采用分层搜索的非自回归（NAR）模型之间的差异。**

序列推荐（即根据用户行为预测下一件物品）与列表续写（即在保持用户整理的列表一致性的同时生成列表）之间仍存在差距。现有的大多数序列推荐模型都被用作下一件物品推荐的召回器，并返回概率最高的前$K$个候选项目。因此，直接将这些模型用于物品列表续写并不合适，因为它们无法捕捉生成列表中物品的顺序及其相互关系。为解决这一问题，一个直接的方法是采用自回归策略，如图 1 左侧所示，逐个生成物品列表。与自然语言处理（NLP）中的语言建模任务 [5] 类似，该方法在全局分类中选取概率最高的物品作为预测结果。现有的列表续写方法 [12, 30] 也基于这一策略设计，但遗憾的是，这种方法难以部署到对效率要求很高的生产环境中。它可能会大幅增加模型推理时间，而该时间会随生成序列的长度线性增长，并随物品词汇表大小呈二次方增长。由于物品不可拆分，推荐系统的物品词汇表远大于自然语言处理中的词元词汇表，因此 softmax 分类器会面临过高的计算复杂度。

**本文工作。** 本文采用非自回归方法，提高物品列表续写的推理效率和推理质量。如图 1 右侧所示，我们通过采用非自回归生成策略同时解码接下来的 $K$个物品，并采用分层搜索方法进行模糊匹配，再进行精确定位，从而弥补先前模型的不足。

具体而言，我们提出了一种**快速 非**自回归**序**列生成模型，即**FANS**，并以双向 Transformer [29]作为骨干网络。具体来说，该模型对典型的自回归 BERT4Rec [25] 模型做出了以下改进。1）不同于自回归模型每次只追加一个 MASK 标记并迭代执行 $K$次，FANS 会一次性拼接 $K$个 MASK 标记，并且只执行一次该过程。2）我们利用物品类别信息（如果没有，可通过简单的分类器获取），并设计了一个两阶段分类器，对潜在表示进行层次化解码。简单来说，两阶段分类器会先预测物品类别，再根据特定类别的物品词汇表预测物品。3）为了提高非自回归生成的推理质量，我们提出使用简单和困难样本，通过课程学习 [1] 来训练模型，以渐进的方式处理这些样本。样本难度根据待预测项目序列的长度来定义，具体内容将在第 4 节介绍。

总而言之，我们的贡献如下： s:

- 我们首次尝试将非自回归生成方法用于物品列表续写，这有助于提升推理效率，并满足工业系统的高要求。
- 我们提出了一种新颖的FANS模型，该模型采用非自回归生成和用于分层搜索的两阶段分类器来提高推理效率，并采用基于课程学习的训练调度器来提升推理质量。
- 我们在四个真实数据集 [12]（包括知乎、Spotify、AotM 和 Goodreads）上对 FANS 模型进行了全面评估，并在一个工业系统中开展了在线 A/B 测试。FANS 在大多数指标上都达到了最先进性能，效率最高提升了 8.7 倍。

## 2 相关工作

**物品列表续写**旨在通过物品序列生成来续写用户整理的列表，同时保持其一致性与连贯性，但对此的研究仍然有限。一类研究聚焦于自动播放列表续接 [2], ，这一方向已受到广泛关注 [7, 28, 31]。然而，这些方法通常专为音乐播放列表设计。

对于通用的物品列表续写，早期研究 [3, 20, 21] 通过马尔可夫链对物品列表建模。近期，CAR [12] 被提出，它基于 SASRec [16]使用一致性感知门控网络对物品一致性特征进行显式建模。HyperTeNet [30] 通过构建用户、物品和列表各自的 $K$-NN 图，利用超图和 Transformer 网络实现个性化列表续写。然而，这些方法以自回归方式生成列表，因此推理效率较低，阻碍了它们在工业系统中的部署。

**近年来，序列推荐** 越来越受欢迎 [13, 23, 26, 35]。相关方法包括基于 CNN 的 Caser[26], 、基于 RNN 的 GRU4Rec [13], 以及近期的基于 Transformer 的方法 [4, 6, 11, 12, 16, 17, 19, 25]。这些序列推荐模型用作下一物品预测的召回器，并返回概率最高的前 $K$ 个候选项目。因此，它们无法直接用于列表续写，因为它们无法捕捉生成列表中的项目顺序及其相互关系。

弥合序列推荐与物品列表续写之间差距的一种直接方法，是在这些推荐模型上采用自回归生成而非物品召回来进行列表生成。然而，由于自回归的特性，这也会带来推理效率较低的问题。

在本文中，我们的 FANS 模型采用非自回归生成，并使用两阶段分类器来提高推理效率。

**非自回归生成**最初在神经机器翻译领域提出，旨在 [8, 9, 18, 33] 提高推理效率，因而在自然语言处理领域受到广泛关注。Mask-Predict [8] 采用条件掩码语言建模任务，迭代地优化生成的序列。Wang 等人 [33] 提出利用辅助正则化来提升推理质量。近期有一些研究 [9]

采用课程学习进行非自回归生成。课程学习模拟人类课程体系中理解序列的过程，训练机器学习模型时从较容易的样本开始，并逐步加入更困难的样本。这有助于提升模型的泛化能力和收敛速度 [32]。

尽管非自回归生成和课程学习已在自然语言处理领域得到探索，我们仍首次尝试将它们用于物品列表续接。此外，由于列表续写的物品词汇表规模（在工业系统中最多可达 240K）远大于自然语言处理领域的词汇表规模（约为 30K），我们进一步设计了一种两阶段分类器，以解决普通分类器因词汇表规模过大而导致的效率低下问题。

## 3 预备知识

符号说明与问题定义。在深入介绍所提出的 FANS 模型细节之前，我们先介绍基本符号，并正式定义物品列表续接问题。设 $\mathrm{V} = \{v_1, v_2, \cdots v_M\}$ 表示物品词汇表，其中 $M$ 为物品数量，且每个物品 $v_i$ 属于类别 $c_{v_i}$，其中 $c_{v_i} \in \mathrm{C} = \{1, 2, \ldots, N\}$ 为类别ID，$N$ 为类别总数。我们用 $\mathrm{W}^j$ 表示类别 $j$ 中的物品集合，并用 $M^j$ 表示 $\mathrm{W}^j$ 的大小。注意，$M = \sum_{j=1}^{N} M^j$。我们用 $(\mathrm{x}, \mathrm{y}) \in \mathrm{D}$ 表示数据集 D 中的一个输入-目标对，其中 $\mathrm{x} = [x_1, x_2, \cdots x_{|\mathrm{x}|}] (x_i \in \mathrm{V})$ 是由匿名用户构建的输入列表，$\mathrm{y} = [y_1, y_2, \cdots y_{|\mathrm{y}|}] (y_i \in \mathrm{V})$ 是目标列表；$|\mathrm{x}|$ 和 $|\mathrm{y}|$ 分别表示输入列表和目标列表的长度。形式上，物品列表续接可定义为：给定输入列表 x，预测作为 x 后续部分的目标列表 y，即通过最大化概率来实现。

$$p\left(\mathrm{y}' = \mathrm{y}|\mathrm{x}\right), \tag{1}$$

其中 $\mathrm{y}'$ 表示长度为 $|\mathrm{y}|$ 的任意可能列表。

**物品分类器**。对于没有类别信息的数据集，我们可以先使用分类器对物品进行分类。分类器可以有多种类型。本文采用一种简单而有效的方法。我们首先使用 word2vec [22] 对物品序列建模并获得物品表示。然后，我们应用 $K$-均值聚类 [10] 将物品划分为 $N$ 个类别。

## 4 所提方法

### 4.1 概述

物品序列建模已在序列推荐领域得到广泛研究
推荐领域中，近期的基于 Transformer 的模型取得了显著成功，例如 SASRec [16],CAR [12], 和 BERT4Rec [25]。虽然 SASRec 和 BERT4Rec 已成功应用于序列推荐系统，但它们被用作召回器，通过建模下一物品预测来获取概率最高的前-$K$ 个候选项目。要解决需要捕捉目标列表中物品顺序及其关系的物品列表续接任务，一种直接的方法是将这些模型用作生成器，并以自回归方式进行序列生成，即预测

$$p\left(\mathrm{y}' = \mathrm{y}|\mathrm{x}\right) = \prod_{i=1}^{|\mathrm{y}|} p\left(y_i' = y_i|\mathrm{x}, \mathrm{y}_{1:i-1}\right). \tag{2}$$

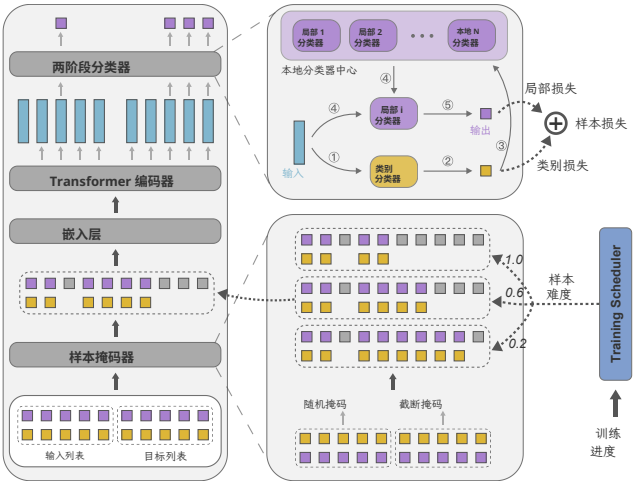



**图 2：FANS模型的框架。紫色和黄色方块分别表示物品令牌和类别令牌，灰色方块表示掩码令牌，蓝色条表示潜在嵌入。被掩码物品的类别令牌设为⟨PAD⟩ 令牌，图中未显示。**

然而，这些自回归方法需要逐个解码物品（通过全局分类器进行下一物品预测），由于推理效率低，阻碍了它们在工业场景中的部署。为满足工业系统对效率的高要求，我们提出了一种名为 FANS 的非自回归方法，用于并行序列生成，即预测

$$p\left(\mathrm{y}' = \mathrm{y}|\mathrm{x}\right) = \prod_{i=1}^{|\mathrm{y}|} p\left(y_i' = y_i|\mathrm{x}\right). \tag{3}$$

图 2 展示了 FANS模型的整体架构，包括四个基本组件，即样本掩码器、嵌入层、Transformer 编码器和两阶段分类器。具体而言，FANS以包含类别信息的输入-目标列表对作为输入。样本掩码器首先处理输入-目标对，对部分物品进行掩码，并生成不同难度级别的令牌级样本。接着，嵌入层将带有稀疏向量的标记序列投影为稠密嵌入序列。然后，Transformer 编码器对嵌入序列进行建模，以学习上下文物品表示。最后，两阶段分类器对输出嵌入进行分层解码，并计算样本损失。

重要的是，为了提高非自回归生成的推理质量，我们采用基于课程学习的训练策略来优化训练。实际上，它是一个训练调度器，用于控制样本掩码器中的样本难度级别。

### 4.2 非自回归生成

为了同时解码整个目标列表，我们采用非自回归生成机制，该机制已在 [9] 神经机器翻译中得到探索和应用。具体而言，我们使用Transformer对连续子序列进行建模，即输入

列表和目标列表，其中目标列表可视为标签。遵循 BERT4Rec [25]，我们采用完形填空 [27] 任务来掩蔽标签序列，并基于上下文信息一次性预测所有物品。

为更好地实现这一目标，我们对目标列表采用截断掩码方案，即仅对列表末尾的子列表进行掩码。同时，为了利用 BERT [5] 预训练所采用的完形填空任务，我们对输入列表应用随机掩码方案，以进一步增强上下文建模。因此，我们将训练目标命名为混合项目掩码。

**SampleMasker.** 样本掩码器依据混合项目掩码处理输入-目标对，用于生成不同难度级别的样本，以训练 FANS模型。

如图 2 所示，每个样本（即一个输入-目标列表对）包含一个物品列表（紫色）和一个类别列表（黄色）。对于前面的物品列表，我们随机掩码其中 $\rho_r$ 比例的物品，并执行以下三种掩码操作之一。对于每个物品 $v_i$：1）以 $\beta_m$ 概率将其替换为特殊标记 MASK；2）以 $\beta_r$ 概率将其替换为词汇表 V $v_r$ 中的随机物品；3）以 $\beta_u$ 概率保持不变，其中 $\beta_m + \beta_r + \beta_u = 1$。对于目标物品列表，我们对末尾 $\rho_t$ 比例的物品进行掩码，并将每个物品替换为特殊的 MASK 标记。同时，输入和目标类别列表将按照以下规则调整：如果在上述过程中对标记 $v_i$ 进行了掩码，则将相应的类别标记 $c_i$ 替换为特殊的 PAD 标记。

此外，我们使用 $\rho_t$ 来定义样本难度：当 $\rho_t$ 接近 0 时，样本较容易；当 $\rho_t$ 接近 1 时，样本较困难。

**嵌入层。** 嵌入层首先构造四种类型的标记序列（即物品、类别、位置和片段标记序列），然后将它们从长度等于词汇表大小的独热向量投影为长度统一的稠密低维向量 $d$（即嵌入维度），最后将四种标记嵌入整合为一个输入嵌入，供 Transformer 编码器使用。

物品标记序列由输入物品列表和目标物品列表拼接而成。通常，我们会在序列开头添加一个特殊的 CLS 标记，并在输入列表和目标列表后添加另一个特殊的 SEP 标记。为简明起见，后续描述中将省略 CLS 和 SEP。拼接后，物品标记序列表示为：

$$\boldsymbol{u} = [x_1, \cdots, \langle \mathsf{MASK} \rangle, \cdots, x_{|\mathbf{x}|}, y_1, \cdots, \langle \mathsf{MASK} \rangle, \cdots \langle \mathsf{MASK} \rangle]. \quad (4)$$

同样，类别标记序列可以构造如下：

$$\boldsymbol{c} = [c_{x_1}, \cdots, \langle \mathsf{PAD} \rangle, \cdots, c_{x_{|\mathbf{x}|}}, c_{y_1}, \cdots, \langle \mathsf{PAD} \rangle, \cdots \langle \mathsf{PAD} \rangle]. \quad (5)$$

遵循 BERT [5]，我们还使用位置标记序列 $[0, 1, 2, \ldots, l-1]$ 来携带不受注意力机制影响的序列位置信息，并使用分段标记序列 $[0, 0, \quad, 0, 1, \quad, 1]$ 来区分输入列表和目标列表，其中 $\boldsymbol{c}$ 、 $\boldsymbol{s}$ 、 $\boldsymbol{u}$ 、 $\boldsymbol{p}$ 的长度相同 $l$。

接下来，我们构建四个连续的标记嵌入矩阵 $\boldsymbol{X}_u \in \mathbb{R}^{(M+2)\times d}$, $\boldsymbol{X}_c \in \mathbb{R}^{(N+2)\times d}$, $\boldsymbol{X}_p \in \mathbb{R}^{l\times d}$ 和 $\boldsymbol{X}_s \in \mathbb{R}^{2\times d}$。注意，MASK 和 PAD 标记将由 $\boldsymbol{X}_u$ 和 $\boldsymbol{X}_c$ 识别，并分别转换为可学习嵌入和固定的零嵌入。

最终的输入嵌入可以如下整合：

$$\boldsymbol{E}^0 = \boldsymbol{A}_u + \boldsymbol{A}_s + \boldsymbol{A}_p + \boldsymbol{A}_s \in \mathbb{R}^{l\times d} \quad (6)$$

其中， $\boldsymbol{A}_u$、 $\boldsymbol{A}_c$、 $\boldsymbol{A}_p$ 和 $\boldsymbol{A}_s$ 分别是物品、类别、位置和分段标记经过转换后的嵌入。

**Transformer 编码器。** 我们的 FANS 模型以 Transformer 网络为骨干网络，由多个 Transformer 层堆叠而成。每个 Transformer 层具有独立的权重，但结构相同。经过多个 Transformer 层编码后，输出序列 $\boldsymbol{E}^H$ 可计算如下：

$$\boldsymbol{E}^h = \mathrm{TRANS}^h\left(\boldsymbol{E}^{h-1}\right), h = 1, 2, \ldots, H, \quad (7)$$

其中， $H$ 是 Transformer 层的数量，$\mathrm{TRANS}^h$ 是第 $h$ 个 Transformer 层。

## 4.3 两阶段分类器

众所周知，掩码语言建模任务采用多层感知机分类器（称为基础分类器），通过 softmax 函数直接推断整个物品词汇表 V 上的概率分布，定义如下：

$$\mathrm{C}_{mlm} : \mathbb{R}^d \to \mathbb{R}^M. \quad (8)$$

随着词汇表不断扩大，计算复杂度和空间复杂度也会相应增加，从而对推理时间产生显著影响。对于物品列表续写，物品词汇表远大于自然语言处理领域所使用的词汇表，因此直接使用基础分类器难以优化且耗时较长。为提高分类效率，一些研究 [15]利用霍夫曼编码算法构建基于频率的二叉树，以进行分层搜索。然而，在物品列表续写场景中，搜索标准应基于内容相似度，而非词频。因此，我们设计了一个两阶段分类器（TSC），通过执行一级分类缩小供二级分类使用的物品搜索空间，从而对潜在嵌入进行分层解码。

具体而言，TSC 模块由一个类别分类器和一个包含 $N$ 个局部分类器的本地分类器中心组成，可表示为：

$$\mathrm{C}_{cat} : \mathbb{R}^d \to \mathbb{R}^N, \quad (9)$$

$$\mathrm{C}^i_{loc} : \mathbb{R}^d \to \mathbb{R}^{M^i}. \quad (10)$$

如图 2 所示，对于样本 $\boldsymbol{t}$ 中掩码标记 $u_k$ 的每个潜在嵌入 $\boldsymbol{E}^H_j$，都会将其输入类别分类器以预测物品类别 $c_{u_k}$，损失函数可定义为：

$$L_{\mathrm{cat}} = -\frac{1}{n}\sum_{i=1}^{n}\sum_{z\in\boldsymbol{t}} \log\left(\mathrm{C}_{cat}\left(\boldsymbol{E}^H_j\right)_{c_{u_k}}\right), \quad (11)$$

其中， $z$ 是样本 $\boldsymbol{t}$ 中的三元组 $\left(u_k, c_{u_k}, \boldsymbol{E}^H_j\right)$（即一个被掩码的物品、其类别及对应的潜在嵌入）， $n$ 是批量大小。如果 $c_{u_k}$ 在所有类别中具有最高概率，本地分类器中心将选择第 $c_{u_k}$ 个局部分类器，将潜在嵌入映射到第 $c_{u_k}$ 个类别

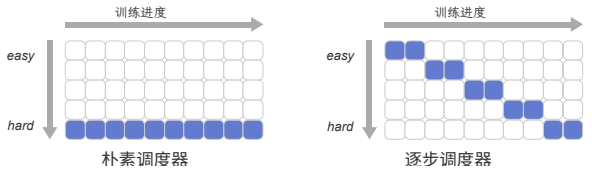



**图 3：朴素调度器与提出的逐步调度器。每个方块表示一个不同难度级别的样本权重，该权重用于训练期间的损失计算。具体而言，在每个训练轮次中，朴素调度器会将所有训练样本转换为困难样本，而逐步调度器会将所有训练样本转换为某个难度级别，并随着训练的进行逐渐从简单变为困难。**

空间 $\mathrm{W}^{c_{u_k}}$。第 $c_{u_k}$个局部分类器的损失函数可表示为：

$$\Phi(z) = 1 - \left| \operatorname{sgn}\left( \arg\max_{i \in (1,\ldots,N)} \mathrm{C}_{cat}\left(\boldsymbol{E}_j^H\right)_i - c_{u_k} \right) \right|, \tag{12}$$

$$L_{\mathrm{loc}}^{c_{u_k}} = -\frac{\sum_{i=1}^{n} \sum_{z \in \boldsymbol{t}} \Phi(z) \log\left( \mathrm{C}_{loc}^{c_{u_k}}\left(\boldsymbol{E}_j^H\right)_{u_k} \right)}{\sum_{i=1}^{n} \sum_{z \in \boldsymbol{t}} \Phi(z) + \epsilon}. \tag{13}$$

其中， $\Phi(z)$是一个布尔检测器，用于判断类别分类器是否给出了正确答案；sgn 是符号函数； $\epsilon$ 是一个很小的实数，用于避免分母为 0。最后，损失函数可表示为：

$$L = L_{\mathrm{cat}} + \frac{1}{N} \sum_{i=1}^{N} L_{\mathrm{loc}}^{i}. \tag{14}$$

## 4.4 课程学习式训练

虽然非自回归生成可以显著提高推理效率，但可能会降低推理质量。为了提高非自回归生成的推理质量，我们设计了一种基于课程学习的训练策略，通过控制样本掩码器中的样本难度级别来实现。

图 3 展示了朴素训练策略和我们提出的逐步训练策略，它们分别对应一个训练调度器。**朴素调度器** 只让样本掩码器在所有训练轮次中生成困难样本批次。因此，样本难度 $\rho_t$始终为 1。**逐步调度器** 采用由易到难的学习方式，要求样本掩码器在不同训练阶段生成具有特定难度级别的样本批次。调度器首先将所有训练轮次划分为等长的单元，每个单元（由若干训练轮次组成）都被视为一个课程学习步数。在每个单元内，样本难度 $\rho_t$保持不变。随着训练进度推进， $\rho_t$逐渐增加，即使用更难的样本来训练 FANS模型。

**表 1：数据集统计信息。**

| | 知乎 | Spotify | AotM | Goodreads |
|---|---|---|---|---|
| #列表 | 18,704 | 72,152 | 12,940 | 15,426 |
| #项目 | 36,005 | 104,695 | 6,264 | 47,877 |
| #类别 | 10 | 10 | 10 | 10 |
| #交互 | 927,781 | 6,809,820 | 162,106 | 1,589,480 |
| 列表中物品平均数量# | 49.59 | 94.38 | 12.53 | 103.04 |
| 列表中物品数量范围# | 10 ~ 200 | 20 ~ 300 | 10 ~ 60 | 20 ~ 300 |
| 密度 | 0.138% | 0.089% | 0.200% | 0.215% |

# 5 实验

## 5.1 实验设置

**数据集。** 根据 [12],，我们在四个真实世界的物品列表续写数据集上开展离线实验，分别为 AotM、知乎、Spotify 和 Goodreads。AotM 数据集由 [21], 提出，其余三个数据集则由 [12]抓取并构建。AotM 和 Spotify 是自动播放列表续接数据集，分别采集自 Art of the Mix[5] 和 Spotify。知乎是一个问答社区，而 Goodreads[6] 是一个图书推荐网站。四个数据集的统计信息汇总于表 1。数据预处理和实现细节见附录。

由于数据集中的物品缺少类别特征，我们根据序列信息对其进行聚类。首先，我们筛除词频小于 10 的物品，并使用连续词袋（CBOW） [22] 模型学习物品潜向量。具体而言，我们将嵌入维度设为 64，窗口大小设为 2。训练目标是根据 $2+2$ 周围的物品预测当前物品。其次，我们使用 $K$-均值聚类算法 [10] 对物品向量进行聚类。我们采用欧几里得距离衡量两个物品嵌入向量之间的距离，并将物品划分为 10 个类别。第三，我们反复执行以下两项操作，直至数据固定：1）从所有列表中移除频次小于 10 的物品；2）根据各数据集的最大和最小长度要求，截断或筛选物品列表。我们将符合条件的列表均匀划分为两段，即输入列表和目标列表。第四，根据最终列表建立物品与类别之间的映射。列表按 8:1:1 的比例划分为训练集、验证集和测试集。

**基线方法与我们方法的变体。** 为验证我们提出的FANS模型的效率和有效性，我们将其与最先进的序列推荐方法和物品列表续接模型进行比较。请注意，由于我们的FANS模型利用了物品类别信息，为了公平比较，我们也为所有对比模型将类别嵌入注入物品嵌入中（如公式6所示，通过向量相加实现）。我们在图4中验证了类别嵌入的有效性。下面简要介绍基线方法和我们方法的变体。

我们与以下自回归模型进行比较：**Caser** [26]利用卷积神经网络（CNN）对局部序列进行垂直和水平建模。**GRU4Rec** [13] 采用

[5]http://www.artofthemix.org/ [6]https://www.goodreads.com/

**表 2：我们的 FANS模型与最先进的方法的比较。蓝色圆圈（ ）表示自回归模型。橙色圆圈（ ）表示采用非自回归生成。绿色圆圈（ ）表示同时采用非自回归生成和所提出的两阶段分类器。“加速比”指标的计算方式为当前模型的延迟（毫秒）与 BERT4Rec 的延迟（毫秒）之比。我们将最佳结果加粗显示，并为次佳结果添加下划线。延迟超过 50 毫秒的结果以红色标示（即 未能满足效率要求），低于 50 毫秒的结果以绿色标示（即 满足效率要求）。**

| | | Caser | GRU4Rec | SASRec | BERT4Rec | CAR | BERT4Rec非自回归 / FANS朴素 | FANS逐步 | FANS时间步-朴素 | FANS时间步-逐步 |
|---|---|---|---|---|---|---|---|---|---|---|
| Zhihu | 归一化折损累计增益@5 | 0.0169 | 0.0077 | 0.0125 | 0.0226 | 0.0156 | 0.0216 | **0.0256** | 0.0192 | 0.0232 |
| | 归一化折损累计增益@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 |
| | 命中率@10 | 0.2960 | 0.2255 | 0.3081 | 0.4275 | 0.3248 | 0.4202 | **0.4819** | 0.3953 | 0.4604 |
| | 延迟（毫秒） | 88.82 | 92.56 | 125.74 | 120.38 | 118.76 | 42.33 | 42.33 | **18.39** | **18.39** |
| | 加速比 | 1.4x | 1.3x | 1.0x | 1.0x | 1.0x | 2.8x | 2.8x | **6.5x** | **6.5x** |
| Spotify | 归一化折损累计增益@5 | 0.0277 | 0.0067 | 0.0081 | 0.0211 | 0.0254 | 0.0272 | 0.0313 | 0.0034 | **0.0315** |
| | 归一化折损累计增益@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 |
| | 命中率@10 | 0.4857 | 0.2110 | 0.2286 | 0.4227 | 0.4755 | 0.5502 | **0.5729** | 0.1188 | 0.5552 |
| | 延迟（毫秒） | 230.51 | 259.18 | 240.48 | 325.17 | 279.07 | 132.88 | 132.88 | **37.46** | **37.46** |
| | 加速比 | 1.4x | 1.3x | 1.4x | 1.0x | 1.2x | 2.4x | 2.4x | **8.7x** | **8.7x** |
| AotM | 归一化折损累计增益@5 | 0.0112 | 0.0053 | 0.0104 | 0.0218 | 0.0072 | 0.0171 | 0.0214 | **0.0229** | 0.0220 |
| | 归一化折损累计增益@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 |
| | 命中率@10 | 0.2923 | 0.1538 | 0.4308 | **0.5538** | 0.2462 | 0.5385 | **0.5538** | 0.4769 | 0.4923 |
| | 延迟（毫秒） | 10.84 | 10.84 | 17.03 | 18.58 | 17.80 | 7.70 | 7.70 | **7.01** | **7.01** |
| | 加速比 | 1.7x | 1.7x | 1.1x | 1.0x | 1.0x | 2.4x | 2.4x | **2.7x** | **2.7x** |
| Goodreads | 归一化折损累计增益@5 | 0.0205 | 0.0074 | 0.0187 | 0.0134 | 0.0232 | 0.0257 | **0.0334** | 0.0276 | 0.0293 |
| | 归一化折损累计增益@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 |
| | 命中率@10 | 0.3294 | 0.1783 | 0.3632 | 0.3379 | 0.4196 | 0.4492 | **0.5149** | 0.4202 | 0.4514 |
| | 延迟（毫秒） | 134.38 | 165.70 | 188.07 | 194.55 | 233.46 | 64.92 | 64.92 | **24.73** | **24.73** |
| | 加速比 | 1.4x | 1.2x | 1.0x | 1.0x | 0.8x | 3.0x | 3.0x | **7.9x** | **7.9x** |

门控循环单元（GRU）网络进行序列推荐。**SASRec** [16] 是一种基于 Transformer 的序列推荐模型，在列表建模中利用单向信息。**BERT4Rec** [25] 是一种最先进的模型，同样以 Transformer 为骨干网络。在训练阶段，它采用自监督掩码语言模型任务来理解上下文信息；在推理阶段，则在输入列表末尾附加 MASK 标记进行预测。**CAR** [12] 是一种基于 SASRec 的 Transformer 物品列表续接模型。它提出了一种一致性感知门控网络，用于显式建模物品一致性特征。由于我们重新生成的数据集中仅存储匿名用户生成列表，因此我们移除了利用用户 ID 特征的 GUPM（通用用户偏好模型）网络。

我们进一步设计了 BERT4Rec 的非自回归版本：**BERT4Rec$_{\text{NAR}}$**（这相当于 FANS 的一个朴素变体，即**FANS$_{\text{NAIVE}}$**）。由于尚无针对物品列表续写提出的非自回归模型，我们通过在列表末尾添加多个 MASK 标记，将 BERT4Rec 模型转换为非自回归模型。

我们评估所提出的FANS模型的以下变体：**FANS$_{\text{STEP}}$** 通过非自回归机制加速并采用逐步调度器。请注意，**FANS$_{\text{NAIVE}}$** 使用朴素调度器。**FANS$_{\text{TSC-NAIVE}}$** 通过非自回归机制和朴素调度器加速，并采用两阶段分类器。**FANS$_{\text{TSC-STEP}}$** 通过非自回归机制和逐步调度器加速，并采用两阶段分类器。

**评估协议**。为了评估推荐列表的质量，我们遵循先前的研究 [24]，采用归一化折损累积增益 [14]（NDCG@k）和命中率（HR@k）指标来衡量各模型的性能。在本研究中，我们设置 $k = \{5, 10\}$。为了评估模型效率，我们测量延迟，即每个样本的推理效率（以毫秒计），通过将推理耗时总和除以测试集中的样本总数得出。我们将批量大小设为 1，并使用配备 NVIDIA GeForce RTX 3090 显卡的设备对五次运行的结果取平均值。

## 5.2 与最先进方法的比较

我们在四个物品列表续接数据集上，从推理质量和效率两方面将 FANS 模型与最先进的方法进行了比较。表 2 汇总了所有模型的整体性能。“加速比”指标计算为当前模型与

**表3：训练调度器对推理质量的影响。实验在知乎数据集上进行。“naive”表示使用朴素调度器。“step=$s$”表示使用包含 $s$ 个课程步骤的逐步调度器。**

| 分类器 | 调度器 | 朴素 / 步=1 | 步=3 | 步=5 | 步=7 | 步=10 |
|---|---|---|---|---|---|---|
| 原始版本分类器 | 归一化折损累计增益@5 | 0.0216 | 0.0232 | 0.0256 | 0.0268 | **0.0274** |
| | 归一化折损累计增益@10 | 0.0325 | 0.0344 | 0.0389 | **0.0390** | 0.0370 |
| | HR@5 | 0.2399 | 0.2734 | **0.2857** | 0.2840 | 0.2820 |
| | 命中率@10 | 0.4202 | 0.4671 | **0.4819** | 0.4792 | 0.4577 |
| 两阶段分类器 | 归一化折损累计增益@5 | 0.0210 | 0.0226 | **0.0232** | 0.0224 | 0.0210 |
| | 归一化折损累计增益@10 | 0.0312 | 0.0329 | **0.0337** | 0.0332 | 0.0310 |
| | HR@5 | 0.2424 | 0.2627 | **0.2670** | 0.2611 | 0.2504 |
| | 命中率@10 | 0.4101 | 0.4362 | **0.4604** | 0.4416 | 0.4201 |

最先进的序列推荐器 BERT4Rec 模型的延迟（毫秒）之比。根据结果，我们可以得出以下观察。

首先，通过比较五种自回归模型（蓝色），我们可以得出结论：基于 Transformer 的 BERT4Rec 模型在推理质量的四项指标（即归一化折损累积增益@{5, 10} 和命中率@{5, 10}）上表现相对出色。具体而言，BERT4Rec 在 AotM 数据集的两项指标上表现最佳，在一项指标上排名第二。

其次，自回归模型（蓝色）的效率和效果都远不及仅通过非自回归机制加速的模型（橙色）。FANS$_{\text{逐步}}$也仅通过非自回归机制加速，因此其延迟与 BERT4Rec$_{\text{非自回归}}$相当。然而，借助逐步课程学习训练策略，它的表现优于 BERT4Rec$_{\text{非自回归}}$，并在 16 种情形中的 13 种达到最先进性能。与 BERT4Rec 相比，FANS$_{\text{逐步}}$不仅在所有四个数据集上都实现了显著加速（加速比为 2.4 倍至 3.0 倍），而且在大多数情形下表现出色。原因可能是：1）以自回归方式生成的结果容易受到先前生成的物品干扰，导致序列偏离正常趋势；2）在基于课程学习的训练调度器帮助下，非自回归方法生成物品的长期趋势预测更加准确。然而，当物品词汇表大小 $M$ 过大时（例如 Spotify 数据集包含 104K 个物品），FANS$_{\text{逐步}}$仍无法满足实时要求（即延迟低于 50 毫秒）。

第三，我们的 FANS模型借助非自回归机制和两阶段分类器（绿色），显著提升了推理效率。与 BERT4Rec$_{\text{NAR}}$ 相比，FANS 利用两阶段分类器对潜在嵌入进行分层解码，从而进一步加速推理。例如，在 Spotify 数据集上，FANS 的加速比可达 8.7 倍，而 BERT4Rec$_{\text{NAR}}$ 的加速比为 2.4 倍。值得注意的是，FANS$_{\text{TSC-NAIVE}}$ 在未采用课程学习时的表现远逊于使用逐步调度器训练的 FANS$_{\text{TSC-STEP}}$。与 FANS$_{\text{STEP}}$ 相比，两阶段分类器显著提高了推理效率。

第四，在小型数据集上，FANS$_{\text{STEP}}$ 足以满足实时推理需求，并能实现最佳推理质量。随着数据集规模增大（尤其是词汇表规模增大），需要考虑权衡而且在大多数情况下，为了提高推理效率，略微牺牲推理质量是可以接受的。

## 5.3 消融研究

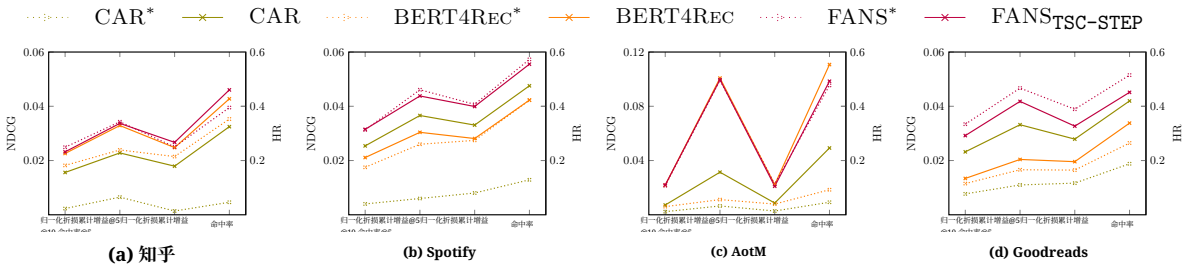
在此，我们研究了不同组件（包括训练调度器和两阶段分类器）对 FANS 性能的影响。表 3 中的实验是在知乎数据集上进行的。我们将课程学习步数参数分别设为 {3，5，7，10}。此外，我们比较了使用 BERT 提出的原始版本分类器 [5] 和我们提出的两阶段分类器时的性能。根据结果，我们可以得出以下观察。**首先**，对于训练调度器，逐步调度器的课程学习步数需要适当调优，否则训练性能会低于朴素调度器。课程学习步数设为 5 时，训练性能最佳。**其次**，通过比较原始版本分类器和两阶段分类器，我们发现，原始版本分类器在大多数指标上的表现优于两阶段分类器。这可能是因为两阶段分类器缩小了物品搜索空间，从而提高了推理效率，但也阻碍了其他类别中的正确物品被选中。

此外，图5比较了基于召回的 BERT4Rec（即 BERT4Rec$_{\text{R}}$）、基于自回归生成的 BERT4Rec，以及基于非自回归生成的 FANS$_{\text{TSC-STEP}}$ 模型的性能。注意，BERT-4Rec$_{\text{R}}$ 与 BERT4Rec 使用相同的训练好的模型，但物品生成方式不同。BERT4Rec$_{\text{R}}$ 作为召回器，仅执行一次下一物品预测操作，利用潜在嵌入检索物品，并将概率最高的 top-$K$ 个候选项目作为一个序列。在对比之下，BERT4Rec 通过自回归生成来获取该序列。尽管 BERT4Rec$_{\text{R}}$ 比 FANS$_{\text{TSC-STEP}}$ 更快，但其推理质量不如 BERT4Rec，且远不及 FANS$_{\text{TSC-STEP}}$。

## 5.4 超参数对推理质量和推理效率的影响

首先，我们研究类别信息对 FANS 性能的影响。CAR$^*$ 和 BERT4Rec$^*$ 分别表示移除 CAR 和 BERT4Rec 的类别特征。由于两阶段分类器基于类别信息，我们使用原始版本的分类器替代它，以构建不注入类别特征的模型，即 FANS$^*$。图4展示了类别特征对三个模型在四个数据集上的推理质量的影响，据此可以得出以下观察结果。首先，通过比较 CAR$^*$ 与 CAR，以及 BERT4Rec$^*$ 与 BERT4Rec，可以看出四个数据集上的推理质量均有所提升，这表明类别信息具有实用价值。其次，在大多数情况下，FANS$^*$ 的表现优于 FANS$_{\text{TSC-STEP}}$ 模型，因为两阶段分类器缩小了物品搜索空间，这可能会导致其他类别中的正确物品无法被选中。

接下来，我们研究可能影响 FANS 推理效率的各种超参数设置。表4展示了物品词汇表大小 $M$和类别词汇表大小 $N$的影响。我们可以得出以下观察结果。首先，值得注意的是，BERT4Rec 的推理时间不会随类别词汇表大小而变化。我们发现，我们的 FANS模型在只有一个

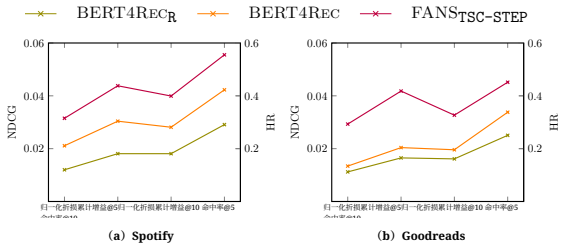



(a) 知乎 (b) Spotify (c) AotM (d) Goodreads

**图4：类别特征的影响。实线表示同时使用物品列表和类别列表训练的模型，虚线表示仅使用物品列表训练的模型。**

**表4：物品词汇表大小（M）和类别词汇表大小（N）对推理效率的影响。**

| | BERT4Rec | FANSTSC | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $N$ | - | 1（原始版本） | 5 | 10 | 20 | 50 | 100 | 250 |
| **知乎**（$M$ = 36，005） | 120.38（1.0 倍） | 41.30（2.9 倍） | 18.96 (6.3x) | 18.81(6.4x) | **17.73 (6.8x)** | 18.46 (6.5x) | 20.50 (5.9x) | 30.82 (3.9x) |
| **Spotify (**$M$ = 104，695**)** | 325.17 (1.0x) | 132.14 (2.5x) | 46.22 (7.0x) | 37.46 (8.7x) | 30.90(10.5x) | 28.14 (11.6x) | **27.33 (11.9x)** | 36.09(9.0x) |
| **AotM**（$M$ = 6，264） | 18.58 (1.0x) | 7.83 (2.4x) | 7.16 (2.6x) | **7.01 (2.7x)** | 8.22 (2.3x) | 9.80（1.9倍） | 12.39（1.5倍） | 22.51（0.8倍） |
| **Goodreads**（$M$ = 47，877） | 194.55（1.0倍） | 64.11（3.0倍） | 29.51（6.6倍） | 24.73（7.9倍） | **21.18（9.2倍）** | 23.22（8.4倍） | 23.42（8.1倍） | 40.50（4.8倍） |




（a）Spotify （b）Goodreads

**图5：序列生成方法（即 FANSTSC-STEP 、BERT4Rec）与基于召回的方法（即 BERT4RecR）之间的性能比较。请注意，BERT4Rec 和 BERT4RecR 使用相同的网络参数，但以不同的方式生成结果：BERT4Rec 以自回归方式续写列表，而 BERT4RecR 则用作召回器，以获取预测概率最高的前 $K$ 个物品。**

类别时，其推理速度始终快于 BERT4Rec 模型；在这种情况下，两阶段分类器与原始版本分类器完全相同，即 FANSTSC-STEP[N=1] 等同于 FANSSTEP 。这表明非自回归模型比自回归模型更高效。其次，在每个数据集上，随着类别规模增大，推理时间会显著减少，随后逐渐增加。这是因为类别规模增大时，第一层类别分类器的耗时会增加，而第二层局部分类器的耗时会减少。当类别规模达到某个阈值时，类别分类器耗时的增加将超过局部分类器耗时的减少。每个数据集的类别规模阈值各不相同，物品词汇表越大，阈值越高。理想情况下，当类别词汇表大小 $N$达到 100 时，我们可以将 Spotify 数据集的处理速度提高 11.9 倍；当 $N$达到 20 时，可以将 Goodreads 数据集的处理速度提高 9.2 倍。

## 5.5 工业系统中的推理效率评估

我们还在华为的音乐列表续接系统中验证了 FANS 的效率，该系统每天为数百万用户提供服务。任务是在用户听完播放列表后提供后续物品列表。由于项目序列较长且音乐数据量巨大（筛选后约有 24 万首），直接使用 3 层 BERT4Rec 模型会导致超过 300 毫秒的难以接受的延迟（在 Tesla P100 PCIe GPU 上测试）。为了提高推理效率，我们采用了 FANSTSC-STEP 模型。我们使用物品分类器对音乐物品进行分类。我们将类别数设为 10（即 FANS[N=10]）和 100（即 FANS[N=100]），并发现推理耗时分别从 300 毫秒降至 42 毫秒（提速 7.1 倍）和 28 毫秒（提速 10.7 倍）。与 FANS[N=10]相比，FANS[N=100] 的推理质量仅低 1.2%，而推理速度快 1.5 倍，这在该应用场景中完全可以接受。

## 6 结论

为了将物品列表续接模型部署到对实时推理有要求的工业场景中，我们提出了 FANS 模型，通过非自回归生成和用于分层解码的两阶段分类器策略来提升推理效率。据我们所知，这是首次尝试将非自回归生成应用于物品列表续写。此外，我们采用基于课程学习的训练策略来训练 FANS 模型，以提升非自回归生成的质量。在真实世界数据集上的实验结果以及在工业系统中的效率评估验证了该模型的优越性

与最先进的方法相比，我们的 FANS 模型能够大幅提升推理效率，同时实现具有竞争力甚至更优的生成质量。

## 致谢

作者谨向来自华为上海研究中心的 Huiling Shen、Jinhua Song 和 Li Jiang 表示感谢，感谢他们提出的宝贵见解，并感谢匿名审稿人提供的有益意见。我们衷心感谢 MindSpore[7] 的支持。MindSpore7 是一个用于本研究的新型深度学习框架。

## 参考文献

[1] Yoshua Bengio、Jérôme Louradour、Ronan Collobert 和 Jason Weston。2009年。课程学习。载于 第*26*届国际机器学习年度会议论文集 （加拿大魁北克省蒙特利尔） （*ICML '09*）。计算机协会，纽约州纽约市，美国，41–48。https://doi.org/10.1145/ 1553374.1553380[2] Ching-Wei Chen、Paul Lamere、Markus Schedl 和 Hamed Zamani。2018年。Recsys Challenge 2018：自动音乐播放列表续写。载于 第*12*届*ACM*推荐系统会议论文集 （加拿大不列颠哥伦比亚省温哥华） （*RecSys '18*）。计算机协会，纽约州纽约市，美国，

527 528 10 1145 3240323 3240342 – 。https://doi.org/ . / . Shuo Chen、[3] Josh L. Moore、Douglas Turnbull 和 Thorsten Joachims。2012年。通过度量嵌入进行播放列表预测。载于 第*18*届*ACM SIGKDD*知识发现与数据挖掘国际会议论文集 （中国北京） （*KDD '12*）。计算机协会，纽约州纽约市，美国，714–722。https://doi.org/ 10 1145 2339530 2339643 . / . Yongjun Chen、Zhiwei [4] Liu、Jia Li、Julian McAuley 和 Caiming Xiong。2022年。用于序列推荐的意图对比学习。载于 *ACM Web Conference 2022*论文集 （虚拟会议，法国里昂） （*WWW '22*）。计算机协会，纽约州纽约市，美国，2172–2182。https: //doi.org/

10 1145 3485447 3512090 . / . Jacob Devlin、Ming-Wei Chang、Kenton [5] Lee 和 Kristina Toutanova。2019年。BERT：用于语言理解的深度双向Transformer预训练。载于*2019*年计算语言学协会北美分会会议：人类语言技术论文集，第*1*卷（长篇与短篇论文）。计算语言学协会，明尼苏达州明尼阿波利斯，4171–4186。https://doi.org/10.18653/v1/N19-1423[6] Ziwei Fan、Zhiwei Liu、Yu Wang、Alice Wang、Zahra Nazari、Lei Zheng、Hao Peng 和 Philip S. Yu。2022年。通过随机自注意力进行序列推荐。载于 *ACM Web Conference 2022*论文集 （虚拟会议，法国里昂） （*WWW '22*）。计算机协会，纽约州纽约市，美国，

2036 2047 10 1145 3485447 3512077 – 。https://doi.org/ . / . Anna [7] Gatzioura、João Vinagre、Alípio Mário Jorge 和 Miquel Sanchez-Marre。2019年。用于改进自动音乐播放列表续写的混合推荐系统。*IEEE*知识与数据工程汇刊 33，5（2019年），1819–1830。[8] Marjan Ghazvininejad、Omer Levy、Yinhan Liu 和 Luke Zettlemoyer。2019年。Mask-Predict：条件掩码语言模型的并行解码。载于*2019*年自然语言处理经验方法会议和第*9*届自然语言处理国际联合会议论文集 （*EMNLP-IJCNLP*）。计算语言学协会，中国香港，

6112 6121 10 18653 1 19 1633 – 。https://doi.org/ . /v /D - [9] Junliang Guo、Xu Tan、Linli Xu、Tao Qin、Enhong Chen 和 Tie-Yan Liu。2020年。通过课程学习微调非自回归神经机器翻译。*AAAI*人工智能会议论文集 34，05（4月，

2020 7839 7846 10 1609 34 05 6289 ），– 。 [10] https://doi.org/ . /aaai.v i 。John A Hartigan 和 Manchek A Wong。 。算法 AS 136：K-means聚类算法。英国皇家统计学会期刊：*C*辑（应用统计）28，1（1979年），100–108。[11] Yun He、Jianling Wang、Wei Niu 和 James Caverlee。2019年。用于推荐用户生成的物品列表的层次化自注意力模型。载于 第*28*届*ACM*信息与知识管理国际会议论文集 （中国北京） （*CIKM '19*）。计算机协会，纽约州纽约市，美国，

1481 1490 10 1145 3357384 3358030 – 。https://doi.org/ . / . [12] Yun He、Yin Zhang、Weiwen Liu 和 James Caverlee。 。面向用户生成的物品列表续写的一致性感知推荐。载于 第*13*届国际网络搜索与数据挖掘会议论文集 （美国德克萨斯州休斯敦） （*WSDM '20*）。计算机协会，纽约州纽约市，美国，250–258。https://doi.org/10.1145/3336191.3371776

[13] Balázs Hidasi、Alexandros Karatzoglou、Linas Baltrunas 和 Do monkos Tikk。2016年。基于循环神经网络的会话推荐。载于第4届国际学习表征会议（ICLR 2016），波多黎各圣胡安，2016年5月2日至4日，会议论文集，Yoshua Bengio 和 Yann LeCun（编）。[14] Kalervo Järvelin 和 Jaana Kekäläinen。2002年。信息检索技术的基于累积增益的评估。《ACM信息系统汇刊》（TOIS）20卷4期（2002年），422–446。[15] Armand Joulin、Edouard Grave、Piotr Bojanowski 和 Tomas Mikolov。2017年。高效文本分类的技巧集。载于计算语言学协会欧洲分会第15届会议论文集：第2卷，短篇论文。计算语言学协会，西班牙瓦伦西亚，427–431。https://aclanthology.org/E

17 2068 - Wang-Cheng Kang 和 Julian McAuley。 [16] 2018年。自注意力序列推荐。载于2018年IEEE国际数据挖掘会议（ICDM）。IEEE，197–206。https://doi.org/10.1109/ICDM.2018.00035[17] Chang Liu、Xiaoguang Li、Guohao Cai、Zhenhua Dong、Hong Zhu 和 Lifeng Shang。2021年。用于序列推荐侧信息融合的非侵入式自注意力。《AAAI人工智能会议论文集》 35, 5 2021 4249 4256 10 1609 35 5 16549 （5月），–。 [18] https://doi.org/。/aaai.v i。Jinglin Liu、Yi Ren、Xu Tan、Chen Zhang、Tao Qin、Zhou Zhao 和 Tie-Yan Liu。2020年。面向非自回归神经机器翻译的任务级课程学习。IJCAI（2020年）。[19] Qijiong Liu、Jieming Zhu、Quanyu Dai 和 Xiaoming Wu。2022年。借助面向新闻推荐的即插即用预训练器提升深度点击率预测。载于第29届国际计算语言学会议论文集。国际计算语言学委员会，2823–2833。[20] Brian McFee 和 Gert RG Lanckriet。2011年。播放列表的自然语言。载于国际音乐信息检索学会（ISMIR），第11卷。537–541。[21] Brian McFee 和 Gert RG Lanckriet。2012年。播放列表方言的超图模型。载于国际音乐信息检索学会（ISMIR），第12卷。343–348。[22] Tomás Mikolov、Kai Chen、Greg Corrado 和 Jeffrey Dean。2013年。向量空间中词表示的高效估计。载于第1届国际学习表征会议（ICLR 2013），美国亚利桑那州斯科茨代尔，2013年5月2日至4日，研讨会论文集，Yoshua Bengio 和 Yann LeCun（编）。[23] Lawrence Rabiner 和 Biinghwang Juang。1986年。隐马尔可夫模型简介。《IEEE ASSP杂志》3卷1期（1986年），4–16。[24] Chuan Shi、Xiaotian Han、Li Song、Xiao Wang、Senzhang Wang、Junping Du 和 S Yu Philip。2019年。异构网络中的多方面信息深度协同过滤。《IEEE知识与数据工程汇刊》33卷4期（2019年），1413–1425。[25] Fei Sun、Jun Liu、Jian Wu、Changhua Pei、Xiao Lin、Wenwu Ou 和 Peng Jiang。2019年。BERT4Rec：基于Transformer双向编码器表征的序列推荐。载于第28届ACM信息与知识管理国际会议论文集（中国北京）（CIKM '19）。计算机协会，美国纽约州纽约市，1441–1450。https://doi.org/

10 1145 3357384 3357895 。/。Jiaxi Tang 和 Ke Wang。个性化 [26] Top-N序列推荐：基于卷积序列嵌入（WSDM '18）。计算机协会，美国纽约州纽约市，565–573。https://doi.org/10.1145/3159652. 3159656[27] Wilson L Taylor。1953年。"完形填空程序"：一种测量可读性的新工具。《新闻季刊》30卷4期（1953年），415–433。[28] Thanh Tran、Renee Sweeney 和 Kyumin Lee。2019年。用于自动播放列表续接的基于对抗马氏距离的注意力歌曲推荐器。载于第42届国际ACM信息检索研究与发展会议论文集（法国巴黎）（SIGIR'19）。计算机协会，美国纽约州纽约市，245–254。https://doi.org/10.1145/3331184.3331234 [29] Ashish Vaswani、Noam Shazeer、Niki Parmar、Jakob Uszkoreit、Llion Jones、Aidan N Gomez、Łukasz Kaiser 和 Illia Polosukhin。2017年。注意力就是一切。《神经信息处理系统进展》30（2017年）。[30] M Vijaikumar、Deepesh Hada 和 Shirish Shevade。2021年。Hypertenet：用于个性化列表续接的超图与Transformer神经网络。载于2021年IEEE国际数据挖掘会议（ICDM）。IEEE，IEEE，1210–1215。[31] Maksims Volkovs、Himanshu Rai、Zhaoyue Cheng、Ga Wu、Yichao Lu 和 Scott Sanner。2018年。用于大规模自动播放列表续接的两阶段模型。载于ACM推荐系统挑战赛2018论文集（加拿大不列颠哥伦比亚省温哥华）（RecSys挑战赛 '18）。计算机协会，美国纽约州纽约市，文章 9, 6 10 1145 3267471 3267480 页码。 [32] https://doi.org/。/。Xin Wang、Yudong Chen 和 Wenwu Zhu。课程学习综述。《IEEE模式分析与机器智能汇刊》（2021年）。[33] Yiren Wang、Fei Tian、Di He、Tao Qin、ChengXiang Zhai 和 Tie-Yan Liu。2019年。带辅助正则化的非自回归机器翻译。《AAAI人工智能会议论文集》33卷01期（2019年7月），5377–5384。

[7]https://www.mindspore.cn

[34] Thomas Wolf、Lysandre Debut、Victor Sanh、Julien Chaumond、Clement Delangue、Anthony Moi、Pierric Cistac、Tim Rault、Rémi Louf、Morgan Funtowicz、Joe Davison、Sam Shleifer、Patrick von Platen、Clara Ma、Yacine Jernite、Julien Plu、Canwen Xu、Teven Le Scao、Sylvain Gugger、Mariama Drame、Quentin Lhoest 和 Alexander M. Rush。2020年。Transformer：自然语言处理领域的最新进展。载于*2020*年自然语言处理实证方法会议论文集：系统演示。计算语言学协会，线上，38–45。
https://www.aclweb.org/anthology/2020.emnlp-demos.6[35] Xiangmin Zhou、Dong Qin、Xiaolu Lu、Lei Chen 和 Yanchun Zhang。2019年。流式在线社交媒体推荐。载于*2019*年第*35*届*IEEE*国际数据工程会议（*ICDE*）。IEEE，IEEE，938–949。

# A 实现细节

## A.1 超参数选择

我们采用 Adam优化器作为梯度下降算法。学习率设为 0.01。四个数据集的批量大小设为 256，嵌入维度设为 64。遵循 BERT [5], ，我们将随机掩码比例 $\rho_r$ 设为 0.15，并将三种掩码操作（即 $\beta_m$、 $\beta_r$、 $\beta_u$）的概率分别设为 0.80、0.10 和 0.10。为确保公平比较，我们将基于 Transformer 的模型的 Transformer 层数以及 GRU4Rec模型的隐藏层数均设为 3。对于 Caser模型，我们遵循其原始实现和设置，并将最大序列长度设为 5。我们将四个数据集上所有基于 Transformer 的方法的注意力头数设为 8。我们将逐步调度器的课程学习步数设为 {3, 5, 7, 10} 之一。除非另有说明，课程步骤数设为 5。

## A.2 推理过程

为推断目标列表，我们执行以下操作：

(1) 模型执行。对于基于非自回归的方法，将长度等于 $|y|$ 的 MASK 标记添加到输入列表$\mathbf{x}$中，训练好的模型一次性生成一个 $|y|\times M$概率矩阵。另一方面，对于基于自回归的方法，该过程会重复 $|y|$ 次：a) 向输入列表$\mathbf{x}$添加一个 MASK 标记，b) 获取当前掩码标记的 $1\times M$概率分布，c) 将概率最高的物品追加到输入列表$\mathbf{x}$ 中以更新输入列表。

(2) 结果去重 . 期望的结果是选择每个位置上概率最高的物品。然而，这可能会导致重复。为解决这一问题，我们在每个位置选择概率最高且未出现在前面位置的物品。这样得到的目标列表长度为 $|y|$，且不包含重复物品。

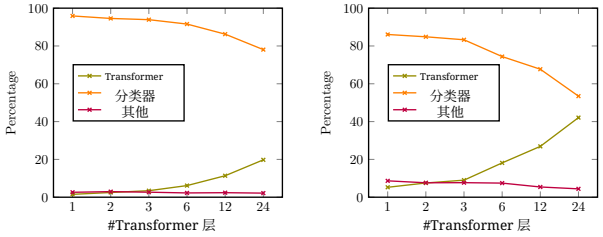
## A.3 其他训练细节

我们基于原始实现实现了 Caser8、SASRec9、GRU4Rec10 和 CAR[11] 。我们使用 Transformers [34]库在 PyTorch 中实现 BERT4Rec 和 FANS模型，源代码和数据可在以下地址获取：https://github.com/Jyonn/ FANS。所有方法均使用配备 24GB 内存的 NVIDIA GeForce RTX 3090 进行训练。结果取 5 次运行的平均值，且 p值小于 0.01。最大训练轮次设为 200，早停的耐心值设为 3。

[8]https://github.com/graytowne/caser_PyTorch [9]
https://github.com/kang205/SASRec [10]
https://github.com/hidasib/GRU4Rec [11]
https://github.com/heyunh2015/ListContinuation_WSDM2020年

# B 推理时间分析

图 6a 展示了非自回归模型（例如第 5 节提到的 BERT4Rec$_{\text{NAR}}$ ）在推理阶段 Transformer 网络与原始版本分类器的耗时占比。可以看出，即使 Transformer 层数较多，基础分类器仍占据总推理时间的主要部分。为缓解这一问题，我们提出利用物品类别信息，并设计两阶段分类器，对潜在表示进行层次化解码。如图 6b 所示，采用两阶段分类器可以大幅缩短推理时间。




**(a) 基础分类器** **(b) 两阶段分类器**

**图 6：非自回归模型（例如 BERT4Rec$_{\text{NAR}}$）中不同模块所占的推理时间百分比。左侧使用 BERT 中的基础分类器，右侧使用我们提出的两阶段分类器。具体而言，基础分类器会计算全局物品词汇表上的概率分布，而我们提出的两阶段分类器则先预测物品类别，再计算该类别对应的局部物品词汇表上的概率分布。分类器（橙色部分）所花费的时间包括获取前-$K$ 个物品的排序操作。**