用于自动播放列表续播的可扩展框架——音乐流媒体服务

Walid Bendada Deezer
Research LAMSADE，巴黎多菲纳大学
PSL research@deezer.com

Guillaume Salha-Galvan
Deezer Research

Thomas Bouabça
Deezer Research

Tristan Cazenave LAMSADE，巴黎多菲纳大学 PSL

arXiv:2304.09061v1 [cs.IR] 12 Apr 2023

摘要

音乐流媒体服务通常会为用户推荐歌曲，以延续他们在这些服务上创建的播放列表。然而，在延续播放列表的同时保留其音乐特征并符合用户偏好，仍是一项颇具挑战性的任务，通常称为自动播放列表续播（APC）。此外，尽管这些服务通常需要从包含数百万候选项的大型曲库中实时选出最佳推荐歌曲，但近期关于 APC 的研究主要集中于可扩展性保证有限的模型，并且只在相对较小的数据集上进行评估。本文提出一个通用框架，用于构建适用于大规模应用、兼具可扩展性与高效性的 APC 模型。该框架基于先表示后聚合策略，从设计上确保可扩展性，同时具备足够的灵活性，可纳入多种表示学习和序列建模技术，例如基于 Transformer 的技术。我们通过在 Spotify 的百万播放列表数据集（MPD）上进行深入的实验验证，展示了该框架的价值；MPD 是规模最大的公开 APC 数据集。我们还介绍了如何在 2022年 利用该框架成功改进 Deezer 生产环境中的 APC。我们报告了该服务大规模在线 A/B 测试的结果，强调了我们的方法在此类真实应用中的实际影响。

CCS 概念

• **信息 系统** →推荐系统； 个性化 化。

关键词

自动播放列表续播、可扩展性、音乐推荐系统、音乐流媒体服务、A/B测试。

ACM 引用格式：

Walid Bendada、Guillaume Salha-Galvan、Thomas Bouabça 和 Tristan Cazenave。2023 年。用于自动播放列表续播的可扩展框架：音乐流媒体服务。载于第*46*届国际*ACM*信息检索研究与发展会议论文集（*SIGIR '23*），*2023*年*7*月*23*日至*27*日，台湾台北。ACM，美国纽约州纽约，11页。https://doi.org/10.1145/3539618.3591628

允许免费制作本作品部分或全部内容的数字副本或纸质副本，用于个人或课堂用途，但不得为营利或商业利益制作或传播副本，且副本首页必须附上此声明和完整引文。必须尊重本作品第三方组件的版权。其他用途请联系所有者/作者。*SIGIR '23*，*2023*年*7*月*23*日至*27*日，台湾台北© 2023年 版权所有者/作者所有。
ACM ISBN 978-1-4503-9408-6/23/07。
https://doi.org/10.1145/3539618.3591628

1 引言

推荐系统对于 Apple Music、Deezer 或 Spotify[6, 24, 46]等音乐流媒体服务正变得越来越重要。虽然这些服务提供了不断扩充的音乐曲库，但其推荐系统通过识别最相关的内容并向每位用户展示，避免了信息过载问题 [3, 45]。推荐系统还使用户能够发现自己可能喜欢的新歌曲、专辑或艺人 [24, 46]。总体而言，人们普遍认为推荐系统是改善这些服务上的用户体验和参与度的有效工具 [3, 5, 38, 62]。

具体而言，音乐流媒体服务通常会推荐歌曲，供用户继续收听他们在这些服务上创建的个人播放列表 。广义而言，播放列表是旨在一起收听的有序歌曲序列，通常符合特定的音乐流派、情绪、文化主题或活动 [2, 5, 60]。在保留其音乐特征并符合用户偏好的同时自动扩充播放列表，仍是一项颇具挑战性的任务，通常称为自动播放列表续播 (APC) [7, 46, 54, 60]。尽管在音乐流媒体服务出现之前，人们就已尝试解决 APC 问题（例如，参见 Bonnin 等人的综述 [4]），但这些服务的兴起推动音乐听众越来越多地消费播放列表 [25], ，此后，对可行 APC 解决方案的需求也持续增长。Schedl 等人 [46]如今将 APC 视为“音乐推荐系统研究中当前最迫切的挑战之一”。

由于 APC 与基于序列和基于会话的推荐密切相关， [8, 56], 近期针对这一问题的研究广泛关注于借鉴已在其他序列建模任务（尤其是语言建模）中取得成功的模型，将其用于推荐。研究人员基于这样一种类比提出了多种有效的 APC 模型：句子中的词语与播放列表中的歌曲相对应。这些模型分别基于 word2vec [52, 55], 循环神经网络 [10, 27],卷积神经网络 [59], 和注意力机制 [19]。然而，正如我们将在第 2 节中详细说明的那样，这些颇具前景的研究往往忽视了可扩展性问题，并且只在真实世界数据集规模相对较小的子集上评估其模型。

然而，对于音乐流媒体服务上的工业应用而言，可扩展性至关重要，因为这些应用必须在包含数千万个候选项的大型曲库中，为APC选择最适合推荐的歌曲。除了训练APC模型时需要处理的歌曲数量之外，推理的可扩展性虽然有时会被忽视，同样至关重要。正如我们将在第3至第5节中进一步阐述的那样，音乐流媒体服务通常必须在线、反复地执行APC，

并且要实时完成。事实上，用户会定期创建新的播放列表并修改现有播放列表。相应地，他们希望系统能够以最小的延迟提供更新后的推荐。

总而言之，近期科学文献中提出的自动播放列表续播模型在复杂性和评估流程方面，与工业应用所要求的可扩展性之间存在差异。作为这一差异的例证，我们特别指出，在 RecSys 2018 自动播放列表续播挑战赛 [26, 60] 中，很少使用现代序列建模技术。为了推动大规模APC领域的研究，这项开放挑战赛采用 Spotify 的百万播放列表数据集（MPD）进行评估；时至今日，该数据集仍是规模最大的公开自动播放列表续播数据集 [7]。为了在这一数据集上提供 APC推荐，许多团队在挑战赛期间倾向于采用相对简单的方法（参见我们在第 2 节中的综述）；截至目前，在如此大规模的场景中应用上述现代自动播放列表续播模型，仍然面临可扩展性方面的挑战。

在本文中，我们提出了一个通用框架，以克服这些挑战并弥合这一已观察到的差异。我们的解决方案在保持足够灵活、能够整合广泛现代方法的同时，还对大规模工业应用中适用的自动播放列表续播解决方案所应满足的要求进行了形式化刻画。更具体地说，本文的贡献如下：

- 我们提出了一种原理严谨的框架，用于构建可扩展且高效的APC模型。该框架基于先表示后聚合策略，能够将各种复杂的APC模型集成到适用于工业应用的大规模系统中。我们的框架将这些模型系统地拆分为两部分：一部分负责歌曲表示学习，另一部分专用于播放列表级序列建模。
- 我们展示了该框架带来的可能性，说明如何设计可扩展的APC模型，使其既能利用最流行的序列建模架构（例如循环神经网络 [22] 或Transformer [53],），又能结合复杂的歌曲表示学习模型（例如处理元数据的神经网络架构 [40]）。
- 我们通过在Spotify的MPD数据集（最大的公开APC数据集）上进行深入的实验验证，展示了该框架对于大规模APC的实证价值。我们还随论文在GitHub上公开发布了源代码，以确保研究结果可复现。
- 我们还介绍了如何在2022年利用该框架改进全球音乐流媒体服务Deezer生产环境中的APC。我们展示了针对该服务用户开展的大规模在线A/B测试结果，强调该框架在此类真实应用中的实际影响。

本文组织如下。第2节将更准确地介绍自动播放列表续播问题，并回顾既有工作。第3节将介绍我们提出的用于构建具备可扩展性且高效的自动播放列表续播模型的框架，详细概述其可能性与局限性。第4节报告我们在公开MPD数据集上的实验设置和结果，第5节介绍我们在Deezer服务上进行的在线A/B测试。最后，我们在第6节总结全文。

2 预备知识

本节首先精确定义我们旨在解决的自动播放列表续播问题，随后回顾相关研究。

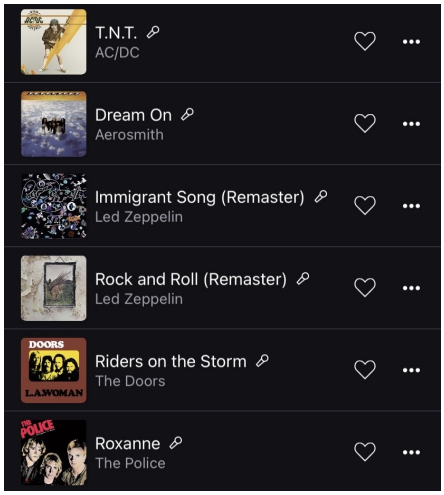


图1：Deezer上用户播放列表的示例。

2.1 问题形式化

2.1.1 符号。 *</style}*本文考虑音乐流媒体服务曲库 S{, ..., s_N} 中的 $N \in \mathbb{N}^*$ 首歌曲。该服务的用户可以创建包含这些歌曲的播放列表，如图 1 所示。用户可以随时更新现有播放列表，添加、移除或重新排列歌曲。我们用 $L \in \mathbb{N}^*$ 表示播放列表的最大长度，这是由服务设定的参数。我们用 P 表示由 S 创建的所有播放列表的集合：

$$\mathcal{P} = \bigcup_{l=1}^{L} \mathcal{S}^l. \tag{1}$$

最后，我们为每首歌曲 $s \in$ S关联一个大小为 $M \in \mathbb{N}^*$的描述性元组 m_s。该元组包含歌曲的元数据信息，例如艺术家的姓名或其所属的专辑：

$$m_s \in \mathcal{M} = \mathcal{M}_1 \times \mathcal{M}_2 \times \cdots \times \mathcal{M}_M. \tag{2}$$

在该公式中，每个M_i 表示给定元数据类型的所有可能取值构成的集合，例如所有可能的艺人构成的集合。

2.1.2 APC。在这种情况下，自动播放列表续播问题在于扩展播放列表 $p \in$ P，添加更多符合 p [7, 46]目标特征的歌曲。形式上，APC模型是一个函数：

$$f\colon \mathcal{P} \times \mathcal{S} \to \mathbb{R}, \tag{3}$$

为每个 $p \in$ P 和 $s \in$ S 关联一个“相关性”或“相似度”得分 $f(p, s)$。得分越高，根据 f， s越有可能适合作为 p的续播内容。为了评估和比较APC模型的性能，标准方法包括：

- 收集一组训练期间未见过的测试播放列表。- 然后，屏蔽这些测试播放列表末尾的歌曲。
- 最后，使用歌曲相关性得分评估各模型补全这些部分屏蔽序列的能力，并计算指标，例如我们将在第4节中讨论的那些指标。

2.2 相关工作

近年来，自动播放列表续播问题受到了研究人员的广泛关注，他们为解决这一问题投入了大量精力。

2.2.1 用于自动播放列表续播问题的协同过滤。 从历史上看，协同过滤（CF） [30, 49] [11, 一34, 46, 54] 2 直是解决自动播放列表续播问题的常用方法。如图 a 所示，用于自动播放列表续播问题的协同过滤方法通常将播放列表数据集表示为稀疏矩阵，其中每一行对应一个播放列表，每一列对应一首歌曲，关联的二值表示则表明该歌曲是否出现在播放列表中。本质上，这些方法旨在利用相似播放列表的内容，推断某首歌曲在一个播放列表中的相关性。其中一些协同过滤方法一方面计算播放列表稀疏向量对之间的内积，另一方面计算歌曲稀疏向量对之间的内积，并利用歌曲在播放列表中的出现信息来计算播放列表与歌曲之间的相似度 [11, 54]。另一些方法假设播放列表-歌曲出现矩阵具有低秩结构，并使用矩阵分解（MF）技术，将其表示为两个矩形稠密矩阵的乘积 [31, 34]。第一个矩阵的维度为 $K \times D$（其中 K 表示数据集中的播放列表数量，另有固定的 $D \ll \min(K, N)$），按行表示播放列表。第二个矩阵的维度为 $D \times N$，按列表示歌曲。它们可以被解释为共同"嵌入"向量空间中播放列表和歌曲的 D 维向量表示。在该空间中，可以计算任意一对对象之间的相似度，例如使用内积 [23]。由于这些协同过滤方法通常无法为不在现有播放列表中的歌曲提供推荐 [18, 40], , 研究人员还提出将元数据信息整合到预测中，其基本假设是：高频歌曲的共现信息应当流向具有相似属性的低频歌曲。整合元数据的协同过滤模型中，具有代表性的例子包括因子分解机（FM） [40]。因子分解机将元数据表示在与播放列表和歌曲相同的嵌入空间中，并将各种元数据嵌入向量之间的内积加到原始预测分数中。它们已成功用于自动播放列表续播问题 [13, 34, 43, 58]。

2.2.2 APC 的序列建模。过去几年，越来越多关于 APC 的研究提出，不再将播放列表数据集表示为稀疏交互矩阵，而是表示为不等长的歌曲序列列表，如图2b所示。这些

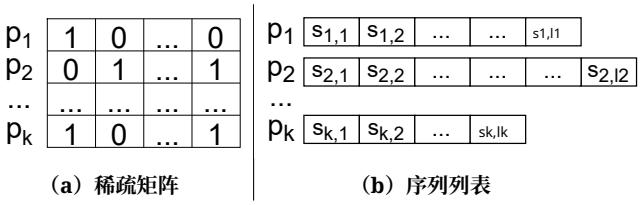
相关研究将已在其他序列建模任务（尤其是语言建模）中取得成功的技术应用于此，以扩展播放列表天然构成的歌曲有序序列 [10, 19, 52, 55, 59] 。

更具体地说，研究者提出了基于 word2vec [52, 55],、循环神经网络 [10, 27],、卷积神经网络 [59], 和注意力机制 [19]的 APC模型。在这些研究的实验评估中，这类序列方法往往优于第 2.2.1 节介绍的纯协同过滤方法。然而，这些令人鼓舞的结果是在真实世界数据集的相对较小子集上取得的[1]（参见表 1 前四行），原因既包括缺少更大型的公开真实世界数据集（对于 MPD 发布 [7]之前的研究），也包括其中一些模型固有的可扩展性问题 [60]。因此，这些方法在涉及数百万首歌曲和播放列表的大规模应用中的实际效果仍有待充分验证。至于最新一类序列建模神经网络架构，

[1]尤其是，可推荐歌曲的范围往往被限制在最热门的歌曲之中。除了可扩展性方面的顾虑，这种限制也让人质疑这些模型能否推荐来自长尾的歌曲。考虑到曲库规模是音乐推荐的特点之一[46]，而且许多模型往往偏向热门艺术家 [32]，这一点尤为重要。

表 1：用于 APC 的公开数据集。

数据集	歌曲	播放列表	交互
AOTM [37]	91 166	27 005	306 830
NOWPLAYING [61]	75 169	75 169	271 177
30音乐 [51]	210 633	37 333	638 933
Melon [14]	649 091	148 826	5 904 718
MPD [7]	2 262 292	1 000 000	66 346 428



（a）稀疏矩阵 （b）序列列表

图2：将APC数据集表示为（a）稀疏的播放列表-歌曲出现矩阵，以及（b）不等长歌曲序列列表。

即 Transformer [53], , 它们已被用于序列化产品推荐（例如 SASRec 和 BERT4Rec [28, 50]），但我们尚未见到其应用于大规模 APC 的案例。同样，近邻技术也已用于序列预测，有时甚至优于复杂的神经模型 [33, 35, 36], , 但尚未在大规模 APC 场景中得到评估。最后，尽管有些方法利用 APC 的多模态特性，另一些方法则关注其序列特性，据我们所知，很少有研究尝试同时显式考虑这两个方面，同时保持足够高的效率以适用于大规模应用。

2.2.3 迈向大规模*APC*。 为促进大规模APC研究，第12届ACM推荐系统会议举办了"RecSys 2018 自动播放列表续播挑战赛" [7], , 该挑战赛基于百万播放列表数据集（MPD）进行评估，随后该数据集向公众发布。该数据集由Spotify用户创建的 - 一百万个播放列表组成。与表1中的其他公开数据集相比，MPD的规模尤为突出，同时也具有较高的数据稀疏性。事实上，尽管数据集中包含超过两百万首歌曲，但其中约60%的歌曲在播放列表中出现的次数不超过两次 [7]。Spotify还提供了歌曲和播放列表的侧信息。在实践中，此类信息通常有助于克服由数据稀疏性引发的冷启动问题 [6, 44]。2019年，Zamani 等人 [60] 分析了该挑战赛的关键洞见，并指出许多团队利用了：

- 集成架构：多个团队结合了若干形式为 $f: \mathrm{P} \times \mathrm{S} \to \mathbb{R}$ 的 APC模型，每个模型都提供不同的"候选"歌曲来扩展播放列表。
- 或者，两阶段架构：首先，一个简单的模型快速为 S 中的所有歌曲打分，并检索出一个"候选"歌曲子集，其规模比原始歌曲集小几个数量级。然后，一个或多个更复杂的模型会对候选歌曲重新排序，以改进 APC推荐。这类两阶段策略的性能高度依赖候选检索模型的质量 [60]。

总体而言，第 2.2.2 节介绍的现代序列建模神经网络很少被用于这些系统中的候选检索，

要么是因为它们的使用方式无法扩展到数百万首歌曲，要么是与替代方案相比效果不佳 [60]。在考察集成／两阶段架构中的候选选择模型，以及直接为所有歌曲打分的单阶段架构时，我们发现，大多数方法都采用了更简单、速度更快的非参数最近邻模型，并将其应用于使用第 2.2.1 节所述模型学习得到的各种基于协同过滤的歌曲／播放列表嵌入表示。总而言之，尽管第 2.2.2 节介绍的方法在较小的数据集上表现出色，但大多数团队还是更青睐简单且可扩展性更强的方法。我们认为，这项挑战凸显了近期科学文献中提出的 APC模型的复杂程度与评估流程，与成功用于大规模应用的简单模型之间的差异。尤其是在普遍质疑深度学习能否真正改善推荐系统的背景下 [12, 36, 41], , 我们认为，APC 研究若能对大规模自动播放列表续接应用中适用解决方案所需满足的要求进行形式化描述，将会受益匪浅。本文余下部分将对此进行阐述。

3 大规模播放列表续播

本节将介绍并分析我们提出的框架，用于构建适用于大规模应用、具备可扩展性且高效的APC模型。

3.1 目标与要求

我们的主要目标是为实现从设计上具备可扩展性的APC模型提供一套全面的指导原则。此外，我们还旨在保持方法的灵活性和通用性。

具体而言，在本文中，我们对可扩展性的定义有两个方面。首先，我们希望构建能够处理包含数百万首歌曲和用户的大型数据集的模型，这通常是音乐流媒体服务实际应用中的要求。此外，工业界从业者的见解强调，可扩展性在推理阶段同样至关重要 [41]。实际上，虽然定期训练APC系统对于纳入新数据（例如，获取新歌曲和播放列表的信息，并相应地更新模型参数）是必要的，但这些更新可以有意延迟，直到观察到一批新事件为止。这样，模型训练操作就可以按照固定计划离线执行，例如每天一次，并使模型能够花费更多时间来更新其参数 [42]。相比之下，推理过程需要反复向用户提供推荐，因此必须完成：

- 在线 在服务中：随着用户定期创建和更新播放列表，APC模型必须扩展的确切歌曲序列无法预先离线处理。
- 延迟极低：在向用户提供更新后的推荐时，尽量降低延迟成本。例如，对于像 Deezer 这样的全球服务，推理时间超过几十毫秒就无法接受。

基于这些原因，在我们的框架中，推理的可扩展性尤为关键。因此，我们将尽量减少复杂度取决于数据集大小的在线操作数量，这些操作由 N, L, M 和 $\max_{1\le k\le M}|\mathbb{M}_k|$ 表示。实际上，歌曲数量 N 是这些参数中最大的。因此，只要条件允许，我们会优先对依赖于 N 的操作进行离线 预计算。

3.2 APC 的可扩展框架

现在，我们正式介绍适用于大规模 APC 的框架。我们重点关注[2] 单阶段模型 $f\colon \mathbb{P}\times\mathbb{S}\to\mathbb{R}$ 直接为每首歌曲 $s\in\mathbb{S}$ 计算相似度分数 $f(p, s)$，以续播某个播放列表 $p\in\mathbb{P}$。此外，我们重点关注[2] 学习用于任意播放列表 p 和歌曲 s的“嵌入”向量表示的 APC 模型。对于某个嵌入维度 $D\in\mathbb{N}^*$，我们分别用$\mathbf{h}_p\in\mathbb{R}^D$ 和$\mathbf{h}_s\in\mathbb{R}^D$表示它们。这类模型可写为：

$$f(p,s)=\mathrm{SIM}(\mathbf{h}_p,\mathbf{h}_s), \tag{4}$$

对于某个相似度函数 $\mathrm{SIM}\colon \mathbb{R}^D\times\mathbb{R}^D\to\mathbb{R}$。对于公式（4）的每个元素，我们遵循第3.1节中的原则，确定框架中必须验证的属性。

3.2.1 歌曲表示。首先，我们注意到，每个歌曲嵌入向量$\mathbf{h}_s$ 取决于歌曲 $s\in\mathbb{S}$，并且可能取决于元数据 $m_s\in\mathbb{M}$。由于曲库中的每首 N歌曲都必须计算一个嵌入向量，因此我们要求提前离线执行此操作，并将结果存储起来，以便通过简单的读取操作快速访问。因此，歌曲嵌入向量不应取决于要扩展的播放列表 p。形式化地说，在我们的框架中，每个$\mathbf{h}_s$ 向量都应通过一个函数[3]计算：

$$\phi\colon \mathcal{S}\to\mathbb{R}^D, \tag{5}$$

为每首歌曲学习唯一的向量表示。

3.2.2 播放列表表示。另一方面，每个播放列表$\mathbf{h}_p$ $\mathbb{P}$ 的表示 $p\in$ 无法提前离线计算，因为用户随时都可以创建和更新播放列表（见第 2.1 节和第 3.1 节）。因此，根据我们的要求，播放列表级别的表示学习操作不应依赖大量输入。我们提出仅根据表征播放列表的歌曲序列（s_{p1}, ..., s_{pl}）来学习$\mathbf{h}_p$ p。由于大规模 APC 通常需要处理稀疏数据 [7], , 我们的框架也旨在尽可能通过参数共享获益，方法是对学习函数$\mathbf{h}_p$施加两项约束：

- 它应将歌曲表示$\mathbf{h}_s$ 作为输入。
- 它应不受播放列表长度影响 $l\in\{1,$ .....因此，通过定义$L\}$. 义 $\mathbb{E}=\bigcup_{l=1}^{L}\mathbb{R}^{D\times l}$，每个$\mathbf{h}_p$ 都应按以下方式计算：

$$g\colon \mathcal{E}\to\mathbb{R}^D, \tag{6}$$

一个函数聚合播放列表中歌曲的表示 p，形成单一的播放列表嵌入表示$\mathbf{h}_p$。

3.2.3 相似度。最终的评分操作 $f(p, s)=\mathrm{SIM}(\mathbf{h}_p, \mathbf{h}_s)$ 用于估计每首歌曲 s 作为某个播放列表的延续是否合适 p，它使用上述 $\mathbf{h}_s$ 和 $\mathbf{h}_p$，构成了 APC 预测过程中的复杂度瓶颈。事实上，假设 APC模型在进行推荐时确实考虑了 $\mathbb{S}$ 中的所有歌曲（而不是某个近似子集），该操作取决于 N。此外，它无法预先离线计算，因为它取决于播放列表表示，而播放列表表示本身是在模型在线观察到需要延续的确切歌曲序列时计算的

[2]这一关注点不失一般性。事实上，源自我们可扩展框架的单阶段 APC 模型可以集成到两阶段／集成系统中，例如第 2.2.3 节所述的系统。此外，尽管嵌入表示在第 2.2.1 节和第 2.2.2 节的相关工作中无处不在，但我们的框架也可以调整，以适用于$\mathbf{h}_p$ 和$\mathbf{h}_s$ 捕获更一般描述性信息的场景。[3]我们将元数据隐含在依赖关系中，因为每个 m_s 都完全依赖于 s。

尽管已有研究提出了基于神经网络的相似度函数 [21]，Rendle 等人 [41]已经证明，这些函数的表现并不优于传统的内积；内积能够快速为数百万个项目评分，因此自然适合在我们的框架中用作相似度函数：$f(p, s) = \mathbf{h}_p, \mathbf{h}_s$。

3.2.4 先表示后聚合。总之，如图3所示，我们的框架要求 APC 模型采用以下结构：

$$f(p,s) = \langle g(\mathbf{h}_{s_{p1}}, ..., \mathbf{h}_{s_{pl}}), \mathbf{h}_s \rangle, \text{ with } \forall k \in \mathcal{S}, \mathbf{h}_k = \phi(k). \quad (7)$$

由于该框架包含一个用于表示歌曲的函数 ϕ 和一个用于聚合这些表示的函数 g，因此下文将其称为先表示后聚合（RTA）。第4节和第5节中的实验将通过实证确认，符合这一框架的各种 APC 模型具有可扩展性。

3.3 RTA架构示例

我们的 RTA 框架在设计上确保了可扩展性，同时仍足够灵活，能够纳入多种现代模型。具体而言，我们的实验将考察以下歌曲表示学习和播放列表级聚合方案。

3.3.1 歌曲表示函数 ϕ。

- $\mathbf{h}_s = \phi_e(s) = \mathbf{e}_s$：我们使用基于播放列表-歌曲共现数据训练的表示学习模型，将每首歌曲 s 直接关联到一个向量 $\mathbf{e}_s \in \mathbb{R}^D$（见第 4.2.1 节）。

- $\mathbf{h}_s = \phi\text{FM}(s) = \sum_{m \in m_s} \mathbf{e}_m$：我们用与歌曲元数据相关联的嵌入向量之和或平均值表示每首歌曲，而这些嵌入向量由基于共现数据训练的模型学习得到。由于该方法与第 2.2.1 节介绍的因子分解机存在联系，我们将其称为 FM。

- $\mathbf{h}_s = \phi\text{NN}(s) = \text{NN}(\{\mathbf{e}_m, \forall m \in m_s\})$：我们使用处理歌曲元数据的神经网络 NN 的输出表示每首歌曲。我们的实验将考察一种类似于 Song 等人所采用的基于注意力机制的神经网络架构 [47]。

3.3.2 播放列表序列建模 / 聚合函数 g。

- $g\text{AVG}(\mathbf{h}_{s_{p_1}}, ..., \mathbf{h}_{s_{pl}}) = \frac{1}{l}\sum_{i=1}^{l} \mathbf{h}_{s_{pi}}$：播放列表表示是其中所包含歌曲的平均表示。- $g\text{CNN}(\mathbf{h}_{s_{p_1}}, ..., \mathbf{h}_{s_{pl}}) = \text{CNN}([\mathbf{h}_{s_{p_1}}; ...; \mathbf{h}_{s_{pl}}])$：门控卷积神经网络（ CNN ）处理歌曲表示的串联，并据此学习播放列表表示。该集成模型的一个关键组成部分赢得了 RecSys 2018 自动播放列表续播挑战赛 [54], 此类门控卷积神经网络在 APC 中的使用，源于其在语言建模中的成功应用 [9]。- $g\text{GRU}(\mathbf{h}_{s_{p_1}}, ..., \mathbf{h}_{s_{pl}}) = \text{GRU}([\mathbf{h}_{s_{p_1}}; ....; \mathbf{h}_{s_{pl}}])$：由门控循环单元（GRU）组成的循环神经网络（RNN）学习播放列表表示。我们的实验将采用与 Hidasi 等人 [22]提出的架构类似的架构，但使用不同的损失函数，该损失函数将在第 3.4 节中介绍。- $g\text{Transformer}(\mathbf{h}_{s_{p_1}}, ..., \mathbf{h}_{s_{pl}}) = \text{解码器}([\mathbf{h}_{s_{p_1}}; ....; \mathbf{h}_{s_{pl}}])$：Transformer 网络的解码器部分 [53] 学习播放列表表示。仅保留解码器用于 APC 的选择，受到 GPT 模型近期兴起的推动；GPT 模型在类似的句子续写任务上展现了最先进的性能 [39]。

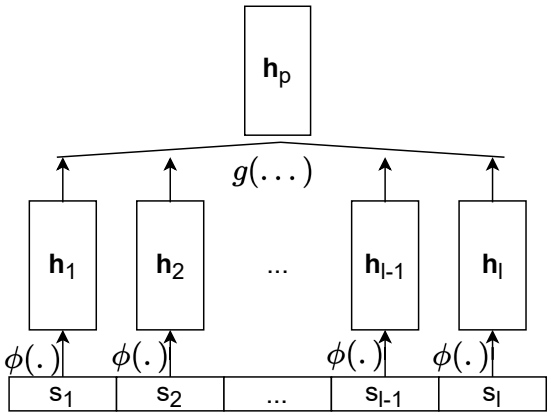


图 3：我们用于 APC 的先表示后聚合 框架。

3.3.3 局限性。 这些示例展示了 RTA框架的多功能性。同时，我们也意识到该框架带来了一些局限性。具体而言，公式（7）无法表达受 word2vec模型启发的 APC 架构 [52, 55]。事实上，这些架构会根据歌曲是用于描述其他歌曲的上下文，还是作为 APC 的候选项进行评估，而采用不同的歌曲表示。采用不同的表示对于语言建模十分重要，因为句子中的词语很少是同义词，因此不应被赋予接近的表示；然而，模型会尝试通过词语的上下文来表示词语，这促使每个词都需要使用不同的表示（参见 Goldberg 等人 [16] 的讨论）。相反，在 APC 的情况下，我们认为出现在同一播放列表中的歌曲应具有接近的表示，即歌曲应与其上下文相似（这是矩阵分解技术 [31]的核心假设）。因此，我们的框架排除了这些方法，转而采用由更多参数共享带来的可扩展性。

3.4 训练流程示例

虽然我们的 RTA 框架具有通用性，但我们建议采用以下训练流程，该流程可用于优化基于此框架构建的所有模型。本文中的实验将采用同一流程来训练所有基于 RTA 的 APC 模型。

我们的初步实验表明，利用加权正则化矩阵分解得到的预训练嵌入向量作为初始歌曲表示，可以提升性能。因此，我们建议首先对所研究的自动播放列表续播问题中的播放列表—歌曲出现矩阵进行分解，并为每首歌曲 s 关联一个用于初始化的 $\mathbf{e}_s \in \mathbb{R}^D$。与此同时，我们通过对具有相同元数据值的歌曲的 $\mathbf{e}_s$ 向量取平均值，来初始化元数据信息的嵌入向量。

从这些初始表示出发，我们使用长度各异的播放列表批次，共同优化选定为 ϕ 和 g 的模型权重。对于长度为 $l \in \{2, \ldots, L\}$ 的每个播放列表 p，我们使用其中前 i 首歌曲创建 $l-1$ 个子播放列表，记作 $p_{:i}$，其中 $i \in \{1, \ldots, l-1\}$。然后，我们从 $S \setminus p$ 中采样一个歌曲集合 $S^-(p)$。由于我们期望 APC模型对 $f(p_{:i}, p_{s_{i+1}})$ 给出较高的相似度分数，而对不在 p 中的歌曲给出较低的分数，因此我们的流程通过梯度下降优化模型权重

通过最小化损失函数 $\mathrm{L}(p) = \mathrm{Lpos}(p) + \mathrm{Lneg}(p)$，实现 [17]：

$$\mathcal{L}_{\mathrm{pos}}(p) = -\sum_{i=1}^{l-1} \log\Big(\frac{1}{1+e^{-f(p_{:i},p_{s_{i+1}})}}\Big), \tag{8}$$

and:

$$\mathcal{L}_{\mathrm{neg}}(p) = -\sum_{i=1}^{l-1} \sum_{s^- \in \mathcal{S}^-(p)} \log\Big(1 - \frac{1}{1+e^{-f(p_{:i},s^-)}}\Big). \tag{9}$$

4 离线实验

现在，我们在百万播放列表数据集（MPD） [7, 60]上对我们的框架进行实验评估。我们已在 GitHub4 上发布源代码，以确保结果可复现，并鼓励今后使用我们的框架。

4.1 实验设置

4.1.1 数据集。 我们的实验使用了完整的 MPD。该数据集包含 Spotify 用户创建的一百万个播放列表，其中使用了两百多万首歌曲（见表 1）。对于每首歌曲 s，我们都有其元数据信息[5]，包括艺术家（art_s）、所属专辑（alb_s）、歌曲时长（dur_s），以及该歌曲在数据集中出现的次数，这一指标称为流行度（pop_s）。因此，$m_s = (\mathrm{art}_s, \mathrm{alb}_s, \mathrm{dur}_s, \mathrm{pop}_s)$。由于 dur_s和 pop_s 是难以直接映射为嵌入向量的数值，我们将相近的数值归入同一分桶：

- 对于时长，我们以每桶 30 秒创建线性分桶，时长超过 20 分钟的歌曲归入同一分桶。我们有：$\mathrm{bucketdur}(s) = \min(40, \left\lfloor \frac{\mathrm{dur}_s}{30} \right\rfloor)$。
- 对于流行度，我们使用对数尺度创建分桶，从而区分较小的数值，而较高的数值则归入同一分桶。令 $\alpha = 45\,000$，即数据集中观测到的最高出现次数，则有：$\mathrm{bucketpop}(s) = \min(100, 1 + 100 \times \left\lfloor \frac{\log(\mathrm{pop}_s)}{\log(\alpha)} \right\rfloor)$。

4.1.2 任务。我们考虑一个大规模APC评估任务，类似于第 2.1.2 节所介绍的任务。具体而言，我们首先从 MPD 中随机抽取 20 000 个长度为 $l \geq 20$ 歌曲的播放列表，构成验证集和测试集，每个集合包含 10 000 个播放列表。剩余的 980 000 个播放列表构成我们的训练集。

接着，我们屏蔽测试播放列表末尾的歌曲，使其仅显示前 n 首种子歌曲。我们考虑了十种不同的配置，n 从 1 到 10 变化，并为每种配置随机选取 1,000 个测试播放列表。我们的实验旨在评估若干 APC模型在 980,000 个训练播放列表上训练后（见第 4.2 节），检索每个测试播放列表中被屏蔽歌曲的能力。与 RecSys 2018 自动播放列表续播挑战赛的做法一致，我们要求每个 APC模型预测一个包含 $n = 500$ 首候选歌曲的排序列表，以续播每个测试播放列表。

4.1.3 指标。下文将分析九种不同的指标，以评估各模型在指定 APC 任务上的表现。首先，我们考察五种以准确性为导向的指标。除了广泛使用的精确率和召回率得分 [46], ，我们还报告了 RecSys 2018 自动播放列表续播挑战赛评估中使用的三种指标 [7]:

- 归一化折损累计增益（NDCG）：用于衡量排序质量。当真实值中的被屏蔽歌曲在 APC模型推荐的候选歌曲排序列表中排名更高时，该指标会提高 [57]。
- R-精确率：综合反映模型推荐的被屏蔽歌曲和艺人的比例。即使预测的歌曲不正确，艺人匹配也会提高该分数 [7]。
- 点击次数：表示在遇到第一首真实值歌曲之前，需要推荐多少批次的十首候选歌曲（从排名前十的候选项开始）。因此，与之前的指标不同，点击次数应当尽量降低 [7]。

Zamani 等人 [60] 提供了这三个指标的确切公式。此外，我们还计算了两个由 Ludewig 和 Jannach [33]描述的面向流行度的分数。这些分数用于监测每个模型在提供推荐时覆盖整个曲库的倾向：

- 覆盖率：计算测试阶段至少被推荐过一次的曲库歌曲所占的百分比。
- 流行度偏差：计算推荐歌曲的平均流行度，具体方法是统计训练集中每首歌曲出现的次数，并应用最小-最大归一化 [33]。分数越低，表示推荐的歌曲流行度越低。

最后，为了比较每个模型的可扩展性，我们计算：

- 训练时间：模型从训练集进行学习，直到在验证集上无法观察到任何准确性相关指标的进一步提升所需的时间。
- 推理时间：为扩展测试播放列表，推荐包含$n\mathrm{reco} = 500$ 首候选歌曲的列表所需的平均时间。

4.2 APC模型

我们比较了十种 APC 模型的可扩展性和性能。

4.2.1 基于我们 *RTA* 框架的模型。首先，我们考虑六种不同的 RTA 模型，它们通过采用以下歌曲表示函数和播放列表聚合/建模函数构建而成：

- MF-AVG[6]：采用表示函数 ϕ_e–MF（见下文），并结合聚合函数gAVG。
- MF-CNN[6]：ϕ_e–MF 与 gCNN 相结合。
- MF-GRU：ϕ_e–MF 与 gGRU 相结合。
- MF-Transformer：ϕ_e–MF 与 gTransformer 相结合。
- FM-Transformer：ϕFM 与 gTransformer 相结合。
- NN-Transformer：ϕNN 与 gTransformer 相结合。

在上述列表中，我们使用第 3.3 节中的记号。此外，我们用 ϕ_{e-MF} 表示表示学习函数 ϕ_e 的一个具体示例。具体而言，ϕ_{e-MF} 使用第 3.4 节中提到的加权正则化矩阵分解初始化歌曲嵌入向量，并通过最小化同一节中定义的损失函数直接优化这些表示。六个模型学习的嵌入向量维度均为 $D = 128$。我们按照第 3.4 节中的流程对它们进行优化，使用随机

[4] https://github.com/deezer/APC-RTA [5]MPD 还包含每个播放列表的标题作为额外信息 [7]。然而，Zamani 等人 [60] 报告称，在 RecSys 2018 自动播放列表续播挑战赛期间，这些信息并未显著提升性能。此外，对于 APC 以及密切相关的任务（例如电台生成/个性化）而言，标题通常不可用 [5]。因此，我们在这些实验中省略了播放列表标题。

[6] 赢得 RecSys 2018 挑战赛的集成系统就包含了这样一种模型 [54]。

表2：在百万播放列表数据集（MPD）上进行的自动播放列表续接（APC）任务，采用了基于我们提出的先表示后聚合（RTA）框架的各种模型及其他基线方法。分数在测试播放列表上计算，并对从1到10变化的 [7], seed取平均值。所有模型都必须为每个测试播放列表提供一个包含$n$reco $= 500$ 首候选歌曲的排序列表，以续接该播放列表。所有嵌入模型学习的表示维度均为$D = 128$，其他超参数按第4.2节所述设置。两列"对*MSS*是否可接受？"表示所报告的训练和推理时间对于音乐流媒体服务（MSS）上的APC是否可接受（✓）或不可接受（×）；相关讨论见第4.3.2节。

模型	精确率 (以 % 计)	召回率 (以 % 计)	R-精确率 (%)	NDCG (以 % 计)	点击次数 (数字)	流行度 (以 % 计)	覆盖率 (以 % 计)	总时间 模型训练	可接受 对于 MSS?	推理时间 按测试播放列表	可接受 对于 MSS?
基线方法											
SKNN	4.93 ± 0.09	36.94 ± 0.49	21.42 ± 0.30	27.66 ± 0.40	3.82 ± 0.22	14.86 ± 0.17	12.64	4 分钟	✓	~ 0.5 秒	×
VSKNN	4.93 ± 0.09	36.83 ± 0.49	21.27 ± 0.30	27.54 ± 0.40	3.84 ± 0.22	15.18 ± 0.17	12.28	4 分钟	✓	~ 0.5 秒	×
STAN	4.32 ± 0.09	32.73 ± 0.48	19.12 ± 0.28	24.26 ± 0.38	4.73 ± 0.23	13.43 ± 0.16	**27.03**	4 分钟	✓	~ 0.5 秒	×
VSTAN	4.59 ± 0.09	34.84 ± 0.49	20.33 ± 0.30	25.99 ± 0.40	4.49 ± 0.23	11.35 ± 0.16	20.54	3 分钟	✓	~ 0.5 秒	×
RTA 模型											
MF-AVG	4.43 ± 0.09	32.64 ± 0.46	19.80 ± 0.28	24.46 ± 0.37	5.95 ± 0.28	21.37 ± 0.19	1.07	15 分钟	✓	< 0.01 秒	✓
MF-CNN	5.00 ± 0.09	37.89 ± 0.45	21.15 ± 0.26	26.83 ± 0.35	3.13 ± 0.19	15.43 ± 0.14	3.99	~ 5 小时	✓	< 0.01 秒	✓
MF-GRU	5.16 ± 0.09	39.16 ± 0.46	21.83 ± 0.27	28.22 ± 0.36	2.77 ± 0.18	12.70 ± 0.13	3.18	~ 7 小时	✓	< 0.01 秒	✓
MF-Transformer	5.20 ± 0.09	39.76 ± 0.47	**22.30 ± 0.28**	29.04 ± 0.38	2.60 ± 0.17	10.46 ± 0.13	5.35	~ 5 小时	✓	< 0.01 秒	✓
FM-Transformer	**5.31 ± 0.09**	**40.46 ± 0.47**	22.29 ± 0.27	**29.21 ± 0.37**	2.52 ± 0.17	11.74 ± 0.13	10.18	~ 6 小时	✓	< 0.01 秒	✓
NN-Transformer	5.26 ± 0.09	40.17 ± 0.47	22.18 ± 0.27	29.14 ± 0.37	**2.17 ± 0.15**	**10.33 ± 0.13**	11.81	~ 6 小时	✓	< 0.01 秒	✓

梯度下降 [17]，每个批次包含 128 个播放列表和 100 个负样本。我们调整所有超参数，以在验证集上最大化 NDCG 分数，并使用 *Optuna* 库 [1] 进行高效的超参数搜索。为简洁起见，我们在 GitHub 仓库 [4] 中报告所有最优超参数值。所有 g卷积神经网络、g门控循环单元、gTransformer 和 ϕNN 模型均包含 1 到 3 层。隐藏层的宽度为 128 到 1024 个单元。g卷积神经网络模型的卷积核大小为 2 到 5。gTransformer 和 ϕNN 的头数为 2 的幂，范围从 2_1 到 2_6。我们测试了从 10^{-3} 到 1 的学习率、从 10^{-9} 到 10^{-4} 的权重衰减，以及从 0 到 0.5 [48] 的丢弃率。对于每个模型，我们在每个 epoch 将学习率减半，并使用验证集 [17]执行早停。我们使用 Python，在一台配备 25 GB 内存和 Tesla P100 GPU 加速器的单 CPU 机器上训练模型。

4.2.2 基线方法。 通过研究这六种不同的 RTA 模型，我们旨在衡量各种现代方法在使用我们的框架时的扩展能力和性能，其中包括此前在大规模 APC 中被忽视的技术。为了建立参照基准，我们还同时评估并比较了四种非 RTA 基线方法：

- SKNN [20]：一种基于会话的最近邻方法，尽管看起来十分简单，但其性能可与卷积神经网络和门控循环单元模型相媲美 [36]。
- VSKNN [33]：SKNN 的一种变体，同时考虑播放列表序列中的歌曲顺序及其流行度。
- STAN [15]：SKNN 的一种变体，可捕捉歌曲在播放列表中的位置、历史播放列表的信息，以及可推荐歌曲在相邻播放列表中的位置。
- VSTAN [36]：将 STAN 和 VSKNN 的理念结合为一种方法，并加入基于序列的物品评分。

我们基于以下两个原因选择了这四种基线方法：

- 首先，就性能而言，在 Ludewig 等人最近的实证分析中，它们往往优于其他方法。 [36]
- 其次，尽管它们是在规模较小的数据集上进行评估的，但我们可以使用现有机器在 MPD 上训练这些模型。

我们通过最大化 NDCG 验证分数来调整所有超参数。我们的超参数候选值集合与 Ludewig 等人 [36] 类似。这四个模型需要每个序列中各项目所执行操作的时间戳。由于 MPD 包含每个播放列表的最新更新时间以及每首歌曲的时长，我们作出了简化假设：每首歌曲都是在上一首歌曲被听过一次后立即添加的。

4.2.3 范围限制。为完整起见，我们指出，我们最初还考虑过其他基线方法，但最终出于可扩展性方面的原因放弃了这些方法。尤其是，我们不会与 Yang 等人的多模态协同过滤模型进行比较。 -

以及 Rubtsov 等人的混合模型。 [43]尽管这两个模型在 RecSys 2018 自动播放列表续播挑战赛中取得了颇具前景的结果，但根据各自的实现，在 MPD 上训练它们都需要超过整整四天 （即使使用配备 28 个核心和 200GB 内存的机器，后一个模型也是如此 [43]）。考虑到 MPD 仅代表 Spotify 用户创建的播放列表中的一小部分，要将这些模型用于此类服务，将需要更多的时间和资源。由于可扩展性是我们工作的核心，因此我们的实验最终舍弃了这些资源需求较高的基线方法。

此外，正如第 3.2 节所述，我们专注于单阶段 APC 模型，直接处理并为 S 中的所有歌曲评分。为清晰呈现并便于比较，我们的实验不包括聚合了多个歌曲评分模型的系统。尽管如此，值得一提的是，实验中的十个模型可以轻松用作两阶段架构中的首个"候选检索"环节（见第 2.2.3 节），也可以作为更大型集成系统的组件（例如，Volkovs 等人 [54] 在挑战赛中将 MF-AVG 和 MF-CNN 结合到他们的集成系统中）。作为未来工作，我们计划拓展本文的研究范围，考察联合使用多个 RTA 模型进行大规模 APC 的方法。

4.3 结果

4.3.1 性能。表2报告了测试播放列表上的所有得分[7]，这些得分是对从1到10变化的n个种子取平均后得到的，并附有95%的置信区间。首先，我们确认了此前的见解 [36]，即经过适当调优的近邻模型（如 SKNN 和 VSKNN）能够取得与某些神经模型相当的性能（例如，SKNN 的归一化折损累计增益为27.66%，MF-CNN 为26.83%，MF-GRU 为28.22%）。

[7]召回率：我们的目标是尽量减少点击次数和流行度。对于覆盖率，我们不报告置信区间，因为它考量的是测试集推荐歌曲的总数。

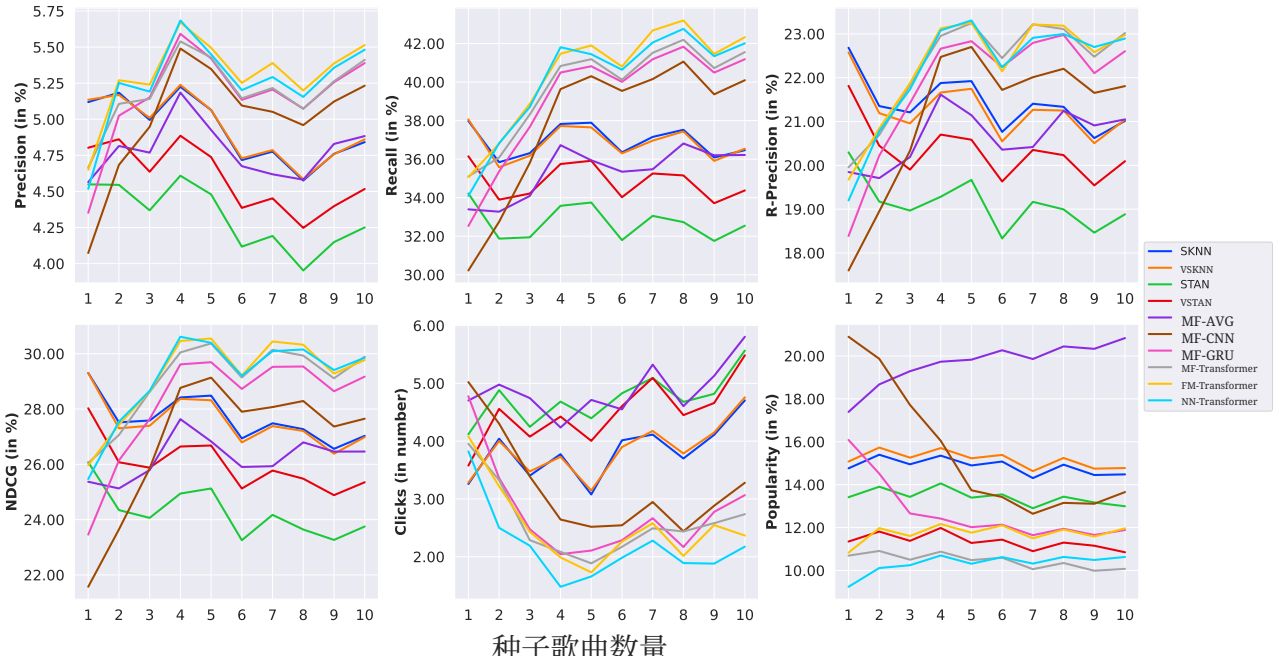


图4：在百万播放列表数据集（MPD）上进行自动播放列表续接（APC）[7]，，使用表2中的设置和模型。与前表不同，我们根据n种子拆分得分，即每个测试播放列表中可见歌曲的数量。

在面向准确率的指标方面，STAN、VSTAN 和 MF-AVG 往往表现欠佳，而使用我们的 RTA框架的基于 Transformer 的模型取得了最佳结果（例如，FM-Transformer 的最高召回率达到40.46%）。利用因子分解机方法进行歌曲表示学习的 FM-Transformer 变体，在三个指标上略微超过了 MF-Transformer 和 NN-Transformer，尽管所有得分在统计上都非常接近。我们特别注意到，基于 Transformer 的模型在排序质量（归一化折损累计增益、点击次数）方面优于其他方案，这一特性对于实际应用十分重要。这验证了我们工作的相关性：我们的工作旨在促进并支持将此类现代序列建模技术引入大规模自动播放列表续接系统。

图4展示了测试播放列表中可见歌曲数量 $n$seed 增加时，不同指标的变化情况。图中不仅印证了此前的结论，还表明序列建模神经网络架构难以处理极短的播放列表。当 $n$seed ≤ 2 时，VSKNN 和 SKNN 甚至优于 Transformer。图4和表2还揭示了关于流行度导向分数的另一项有趣观察。平均而言，MF-AVG 给出的推荐最为主流（流行度评分最高，为 21.37%；曲库覆盖率最低，仅为 1.07%），其次是 VSKNN 和 SKNN。相反，Transformer 的出色表现并非源于只关注热门歌曲。有趣的是，STAN 和 VSTAN 推荐的歌曲最多样化（例如，STAN 的覆盖率最高，达到 27.03%），但代价是性能下降。

4.3.2 可扩展性。现在我们来讨论可扩展性指标。这些指标最能体现我们的 RTA框架 对大规模工业应用的价值。虽然基线方法的训练速度更快（在我们的机器上需要 3 到 4 分钟），但基于 RTA 的模型训练时间为 15 分钟到几小时，对于音乐流媒体服务上的 APC 来说仍然可以接受。事实上，如第3.1节所述，此类服务无需实时执行训练操作。可以按固定计划（例如每天一次）离线执行训练，以更新生产模型的参数。实际上，表2中评估的所有模型都可以在一夜之间整合新的输入数据。

相比之下，音乐流媒体服务无法延迟推理过程。如第3.1节所述，为了向众多用户实时提供推荐，该过程通常需要反复在在线环境中执行。具体而言，我们在前一节中解释过，对于 Deezer 这样的服务，在生产环境中，每个播放列表的推理时间若超过几十毫秒，通常是不可接受的。在表2中，基线方法需要500毫秒（0.5秒），才能为每个测试播放列表扩展出一个包含500首推荐歌曲的列表。所有 RTA 模型完成相同操作所需的时间都不到10毫秒，即使是最先进的模型 NN-Transformer 也是如此；该模型使用基于注意力机制的网络进行歌曲表示，并使用 Transformer 进行播放列表建模。虽然所有模型都能从更好的硬件中受益，但总体而言，我们的 RTA 模型所需的推理时间使其更适合用于生产环境。这些结果凸显了我们的框架将复杂但前景广阔的模型整合到有效系统中的能力；对于大规模APC而言，这些模型此前常被忽视，而如今能够满足现实世界的工业应用所要求的可扩展性要求。

5 在线实验

本节展示了我们的框架近期如何帮助 Deezer 大规模改进 APC。本节的目标并不是重新评估第4节中的所有模型，而是通过进一步说明我们的框架在工业环境中成功应用所体现的实际价值，来补充我们的分析。

5.1 音乐流媒体服务上的 APC

全球音乐流媒体服务 Deezer 提供 APC 功能，如图5所示。该功能允许数百万 Deezer 用户自动续播他们在该服务上创建的播放列表。2022年4月，我们针对这一功能进行了一项大规模在线A/B测试，旨在利用 RTA 框架带来的可能性来改进推荐。在测试之前，我们在生产环境中使用的参考系统（下文称为“Reference”）采用了类似于 MF-AVG 的协同过滤模型。此外，该系统还采用了多种内部规则，以控制推荐歌曲列表中艺术家和专辑出现的频率。

测试期间，我们改用基于 RTA 的 MF-Transformer（第4节中的三个最佳模型之一），为随机选取的一组测试用户提供 APC。我们使用包含 2500 万个用户播放列表的私有数据集训练 MF-Transformer 和 Reference。两个模型都从包含 200 万首可推荐歌曲的候选池中选择推荐歌曲。如图5所示，用户请求续播播放列表时，会看到一个可垂直滚动的列表，其中有 100 首歌曲；最前面的五首歌曲会首先显示在移动屏幕上。用户并不知道算法已发生变化。

5.2 在线 A/B 测试结果

首先，我们的 RTA 框架证明了它在将 MF-Transformer 集成到 Deezer 生产环境中是一项宝贵的资产。它成功集成了该模型，使其能够以极低的延迟成本为用户执行 APC。举例来说，我们观察到服务上的第99百分位推理时间仅为 12 毫秒，用户几乎无法察觉（使用配备十个线程的 AMD EPYC 7402P CPU 机器，每个推理过程计算播放列表表示、为两百万首歌曲评分并返回 100 条推荐，耗时即为此时间）。这一示例展示了我们的框架如何帮助从业者

有效利用现代自动播放列表续播模型，例如 Transformer；否则，这些模型可能会因可扩展性问题而无法投入使用。

在性能方面，我们的测试证实，MF-Transformer 在 Deezer 上优于参考组。使用 MF-Transformer 提升了 APC 功能的内部性能指标，例如添加到播放列表率，即用户将推荐歌曲添加到播放列表的比例。出于保密原因，本文不报告确切的添加到播放列表率，也不公布各组所涉及的用户数量。相反，图 6 展示了相对于参考组的相对 添加到播放列表率。平均而言，接触到 MF-Transformer 的用户添加到播放列表中的推荐歌曲比参考组用户多 70%。

在完成这项最终测试后，MF-Transformer 已部署给所有用户。未来的实验将评估该服务上的其他 APC 模型，例如 FM-Transformer 和 NN-Transformer。此外，我们计划研究如何将该框架应用于 Deezer 上的相关序列任务，包括电台个性化。

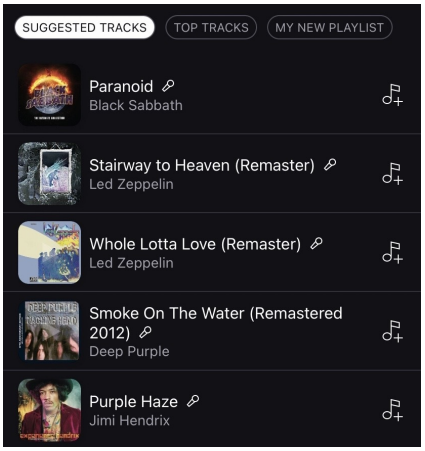


图5：Deezer 上的 APC：我们的 MF-Transformer 推荐的歌曲，用于延续图1中的播放列表。

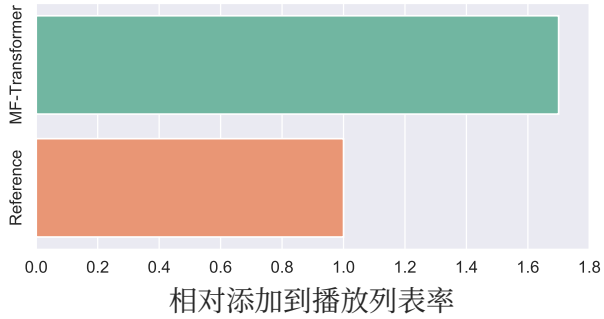


图6：在线 A/B 测试：与 Deezer 的参考模型相比，相对的添加到播放列表率。差异在 1% 水平上具有统计显著性（p值 < 0.01）。

最后，鉴于近期研究 [29] 发现了播放列表数据集中的偏差，而这些偏差可能被推荐系统复现，我们打算在该框架的背景下进一步研究这些重要方面。

6 结论

本文介绍了一个通用框架，用于构建可扩展的APC模型，以满足大规模工业应用（例如音乐流媒体服务）所需的预期要求。我们详细概述了该框架的可能性与局限性，展示了其在整合广泛的先进表示学习和序列建模技术方面的灵活性；由于这些技术较为复杂，它们在以往的大规模APC实验中往往被忽视。我们通过针对最大的公开APC数据集开展离线实验，以及通过在线实验在全球音乐流媒体服务上大规模改进APC，证明了该框架的实证价值。除前文讨论的在同一服务上开展的后续测试外，我们未来还将研究多模态歌曲表示学习模型（例如，除使用数据外，还处理音频信号和歌词），以增强框架中的表示部分。我们还将致力于使其能够实时适应用户反馈。

参考文献

[1] Takuya Akiba、Shotaro Sano、Toshihiko Yanase、Takeru Ohta 和 Masanori Koyama。2019年。Optuna：新一代超参数优化框架。《第25届ACM SIGKDD国际知识发现与数据挖掘会议论文集》（2019年），2623–2631。[2] Walid Bendada、Guillaume Salha 和 Théo Bontempelli。2020年。利用上下文多臂老虎机实现音乐流媒体应用中的轮播个性化。《第十四届ACM推荐系统会议论文集》（2020年），420–425。[3] Jesús Bobadilla、Fernando Ortega、Antonio Hernando 和 Abraham Gutiérrez。2013年。推荐系统综述。《Knowledge-Based Sys.》46（2013年），109–132。[4] Geoffray Bonnin 和 Dietmar Jannach。2014年。音乐播放列表的自动生成：综述与实验。《Comput. Surveys》47，2（2014年），1–35。[5] Théo Bontempelli、Benjamin Chapus、François Rigaud、Mathieu Morlon、Marin Lorant 和 Guillaume Salha-Galvan。2022年。Flow Moods：在Deezer上按情绪推荐音乐。《第16届ACM推荐系统会议论文集》（2022年），452–455。[6] Léa Briand、Guillaume Salha-Galvan、Walid Bendada、Mathieu Morlon 和 Viet-Anh Tran。2021年。用于音乐流媒体应用用户冷启动推荐的半个性化系统。《第27届ACM SIGKDD知识发现与数据挖掘会议论文集》（2021年），2601–2609。[7] Ching-Wei Chen、Paul Lamere、Markus Schedl 和 Hamed Zamani。2018年。RecSys挑战赛2018：自动音乐播放列表续播。《第12届ACM推荐系统会议论文集》（2018年），527–528。[8] Tran Khanh Dang、Quang Phu Nguyen 和 Van Sinh Nguyen。2020年。基于深度学习的基于会话的推荐系统方法研究。《SN Computer Science》1，4（2020年），1–13。[9] Yann N Dauphin、Angela Fan、Michael Auli 和 David Grangier。2017年。利用门控卷积网络进行语言建模。《第34届国际机器学习会议论文集》（2017年），933–941。[10] Tim Donkers、Benedikt Loepp 和 Jürgen Ziegler。2017年。基于序列用户的循环神经网络推荐。《第11届ACM推荐系统会议论文集》（2017年），152–160。[11] Guglielmo Faggioli、Mirko Polato 和 Fabio Aiolli。2018年。用于播放列表续播任务的高效基于相似度的方法。《ACM推荐系统挑战赛2018论文集》（2018年），1–6。[12] Maurizio Ferrari Dacrema、Paolo Cremonesi 和 Dietmar Jannach。2019年。我们真的取得了很大进展吗？对近期神经推荐方法的令人担忧的分析。《第13届ACM推荐系统会议论文集》（2019年），101–109。[13] Andres Ferraro、Dmitry Bogdanov、Jisang Yoon、KwangSeob Kim 和 Xavier Serra。2018年。利用结合文本和音频特征的混合推荐系统实现自动播放列表续播。《ACM推荐系统挑战赛2018论文集》（2018年），1–5。[14] Andres Ferraro、Yuntae Kim、Soohyeon Lee、Biho Kim、Namjun Jo、Semi Lim、Suyon Lim、Jungtaek Jang、Sehwan Kim、Xavier Serra 等。2021年。Melon播放列表数据集：用于基于音频的播放列表生成和音乐标签提取的公开数据集。《2021年IEEE国际声学、语音与信号处理会议论文集》（2021年），536–540。[15] Diksha Garg、Priyanka Gupta、Pankaj Malhotra、Lovekesh Vig 和 Gautam Shroff。2019年。面向基于会话的推荐的序列与时间感知邻域：STAN。《第42届国际ACM SIGIR信息检索研究与开发会议论文集》（2019年），1069–1072。[16] Yoav Goldberg 和 Omer Levy。2014年。word2vec详解：推导Mikolov等人的负采样词嵌入方法。arXiv:1402.3722（2014年）。[17] Ian Goodfellow、Yoshua Bengio 和 Aaron Courville。2016年。《深度学习》。MIT出版社。[18] Jyotirmoy Gope 和 Sanjay Kumar Jain。2017年。推荐系统中冷启动问题解决方法综述。《2017年计算、通信与自动化国际会议论文集》（2017年），133–138。[19] Lei Guo、Hongzhi Yin、Qinyong Wang、Tong Chen、Alexander Zhou 和 Nguyen Quoc Viet Hung。2019 年。流式基于会话的推荐。《第 25 届 ACM SIGKDD国际知识发现与数据挖掘会议论文集》（2019年），1569–1577。[20] Negar Hariri、Bamshad Mobasher 和 Robin Burke。2015年。交互式推荐中对用户偏好变化的适应。《第24届国际人工智能联合会议论文集》（2015年）。[21] Xiangnan He、Lizi Liao、Hanwang Zhang、Liqiang Nie、Xia Hu 和 Tat-Seng Chua。2017年。神经协同过滤。《第26届国际万维网会议论文集》（2017年），173–182。[22] Balázs Hidasi、Alexandros Karatzoglou、Linas Baltrunas 和 Domonkos Tikk。2016年。利用循环神经网络进行基于会话的推荐。《第4届国际学习表示会议论文集》（2016年）。[23] Yifan Hu、Yehuda Koren 和 Chris Volinsky。2008年。针对隐式反馈数据集的协同过滤。《第8届IEEE国际数据挖掘会议论文集》（2008年），263–272。[24] Kurt Jacobson、Vidhya Murali、Edward Newett、Brian Whitman 和 Romain Yon。2016年。Spotify上的音乐个性化。《第10届ACM会议推荐系统研讨会（RecSys 2016）》。[25] Jack Joyce。2016年。播放列表的听众数量超过专辑，Loop研究称。发表于“音乐商业协会”的博客文章：https://musicbiz.org/news/playlists-overtake-albums-listenership-says-loop-study/（2016年）。[26] Dietmar Jannach、Gabriel de Souza P. Moreira 和 Even Oldridge。2020年。为什么深度学习模型尚未持续赢得推荐系统竞赛？一篇立场论文。2020年推荐系统挑战赛论文集（2020年），44–49。[27] How Jing 和 Alexander J Smola。2017年。神经生存推荐器。第10届ACM国际网络搜索与数据挖掘会议论文集（2017年），515–524。[28] Wang-Cheng Kang 和 Julian McAuley。2018年。自注意力序列推荐。2018年IEEE国际数据挖掘会议论文集（2018年），197–206。[29] Peter Knees、Andres Ferraro 和 Moritz Hübler。2022年。音乐推荐中的偏差与反馈回路：唱片公司影响研究。载于多目标推荐系统研讨会（MORS'22），与第16届ACM推荐系统会议联合举办，第22卷。2022年。[30] Yehuda Koren 和 Robert Bell。2015年。协同过滤的进展。推荐系统手册（2015年），77–118。[31] Yehuda Koren、Robert Bell 和 Chris Volinsky。2009年。推荐系统的矩阵分解技术。Computer 42，8（2009年），30–37。[32] Dominik Kowald、Markus Schedl 和 Elisabeth Lex。2020年。音乐推荐中流行度偏差的不公平性：一项可复现性研究。第42届欧洲信息检索会议论文集（2020年），35–42。[33] Malte Ludewig 和 Dietmar Jannach。2018年。基于会话的推荐算法评估。用户建模与用户自适应交互 28，4（2018年），331–390。[34] Malte Ludewig、Iman Kamehkhosh、Nick Landia 和 Dietmar Jannach。2018年。有效的最近邻音乐推荐。ACM推荐系统挑战赛2018论文集（2018年），1–6。[35] Malte Ludewig、Noemi Mauro、Sara Latifi 和 Dietmar Jannach。2019年。基于会话的推荐中神经网络方法与非神经网络方法的性能比较。第13届ACM推荐系统会议论文集（2019年），462–466。[36] Malte Ludewig、Noemi Mauro、Sara Latifi 和 Dietmar Jannach。2021年。基于会话的推荐算法的实证分析。用户建模与用户自适应交互 31，1（2021年），149–181。[37] Brian McFee 和 Gert Lanckriet。2012年。播放列表方言的超图模型。第13届国际音乐信息检索学会会议论文集（2012年），343–348。[38] Ruihui Mu。2018年。基于深度学习的推荐系统综述。IEEE Access 6（2018年），69009–69022。[39] Alec Radford、Karthik Narasimhan、Tim Salimans、Ilya Sutskever 等。2018年。通过生成式预训练改进语言理解。OpenAI（2018年）。[40] Steffen Rendle。2010年。因子分解机。2010年IEEE国际数据挖掘会议论文集（2010年），995–1000。[41] Steffen Rendle、Walid Krichene、Li Zhang 和 John Anderson。2020年。神经协同过滤与矩阵分解再探。第十四届ACM推荐系统会议论文集（2020年），240–248。[42] Francesco Ricci、Lior Rokach 和 Bracha Shapira。2011年。推荐系统手册引言。推荐系统手册（2011年），1–35。[43] Vasiliy Rubtsov、Mikhail Kamenshchikov、Ilya Valyaev、Vasiliy Leksin 和 Dmitry I Ignatov。2018年。用于自动播放列表续播的混合式两阶段推荐系统。ACM推荐系统挑战赛2018论文集（2018年），1–4。[44] Guillaume Salha-Galvan、Romain Hennequin、Benjamin Chapus、Viet-Anh Tran 和 Michalis Vazirgiannis。2021年。利用受重力启发的图自动编码器进行冷启动相似艺术家排名。第15届ACM推荐系统会议（2021年），443–452。[45] Markus Schedl。2019年。音乐推荐系统中的深度学习。应用数学与统计学前沿（2019年），44。[46] Markus Schedl、Hamed Zamani、Ching-Wei Chen、Yashar Deldjoo 和 Mehdi Elahi。2018年。音乐推荐系统研究当前的挑战与展望。国际多媒体信息检索期刊 7，2（2018年），95–116。[47] Weiping Song、Chence Shi、Zhiping Xiao、Zhijian Duan、Yewen Xu、Ming Zhang 和 Jian Tang。2019年。AutoInt：通过自注意力神经网络进行自动特征交互学习。第28届ACM国际信息与知识管理会议论文集（2019年），1161–1170。[48] Nitish Srivastava、Geoffrey Hinton、Alex Krizhevsky、Ilya Sutskever 和 Ruslan Salakhutdinov。2014年。Dropout：一种防止神经网络过拟合的简单方法。机器学习研究期刊 15，1（2014年），1929–1958。[49] Xiaoyuan Su 和 Taghi M Khoshgoftaar。2009 年。协同过滤技术综述。人工智能进展 2009（2009年）。[50] Fei Sun、Jun Liu、Jian Wu、Changhua Pei、Xiao Lin、Wenwu Ou 和 Peng Jiang。2019年。BERT4Rec：基于Transformer双向编码器表示的序列推荐。第28届ACM国际信息与知识管理会议论文集（2019年），1441–1450。

[51] Roberto Turrin、Massimo Quadrana、Andrea Condorelli、Roberto Pagano 和 Paolo Cremonesi。2015年。30Music聆听与播放列表数据集。*RecSys*海报（2015年），75。[52] Flavian Vasile、Elena Smirnova 和 Alexis Conneau。2016年。Meta-Prod2vec：利用侧信息进行推荐的产品嵌入。第*10*届*ACM*推荐系统会议论文集（2016年），225–232。[53] Ashish Vaswani、Noam Shazeer、Niki Parmar、Jakob Uszkoreit、Llion Jones、Aidan N Gomez、Łukasz Kaiser 和 Illia Polosukhin。2017年。Attention Is All You Need。神经信息处理系统进展 30（2017年）。[54] Maksims Volkovs、Himanshu Rai、Zhaoyue Cheng、Ga Wu、Yichao Lu 和 Scott Sanner。2018年。面向大规模自动播放列表续播的两阶段模型。*ACM*推荐系统挑战赛*2018*论文集（2018 年），1–6。[55] Dongjing Wang 、 Shuiguang Deng 、 Xin Zhang 和 Guandong Xu。2016年。利用元数据学习音乐嵌入，以实现上下文感知推荐。*2016*年*ACM*国际多媒体检索会议论文集（2016年），249–253。[56] Shoujin Wang、Liang Hu、Yan Wang、Longbing Cao、Quan Z Sheng 和 Mehmet Orgun。2019年。序列推荐系统：挑战、进展与展望。第*28*届国际人工智能联合会议论文集（2019年）。

[57] Yining Wang、Liwei Wang、Yuanzhi Li、Di He、Wei Chen 和 Tie-Yan Liu。2013年。NDCG排序指标的理论分析。载于 第*26*届学习理论会议，第8卷。6。[58] Hojin Yang、Yoonki Jeong、Minjin Choi 和 Jongwuk Lee。2018年。MMCF：用于自动播放列表续播的多模态协同过滤。*2018*年*ACM*推荐系统挑战赛论文集（2018），1–6。[59] Fajie Yuan、Alexandros Karatzoglou、Ioannis Arapakis、Joemon M Jose 和 Xiangnan He。2019年。用于下一项目推荐的简单卷积生成网络。第*12*届*ACM*国际网络搜索与数据挖掘会议论文集（2019），582–590。[60] Hamed Zamani、Markus Schedl、Paul Lamere 和 Ching-Wei Chen。2019年。对2018年ACM推荐系统挑战赛中自动音乐播放列表续播所采用方法的分析。*ACM*智能系统与技术汇刊（*TIST*）10，5（2019），1–21。[61] Eva Zangerle、Martin Pichl、Wolfgang Gassler 和 Günther Specht。2014年。#now-playing音乐数据集：从Twitter中提取聆听行为。第*1*届互联网规模多媒体管理国际研讨会论文集（2014），21–26。[62] Shuai Zhang、Lina Yao、Aixin Sun 和 Yi Tay。2019年。基于深度学习的推荐系统：综述与新视角。计算机调查 52，1（2019），1–38。