PinnerSage：Pinterest的多模式用户嵌入推荐框架

Aditya Pal*, Chantat Eksombatchai*, Yitong Zhou*, Bo Zhao, Charles Rosenberg, Jure Leskovec
Pinterest公司。
{apal,pong,yzhou,bozhao,crosenberg,jure}@pinterest.com

ABSTRACT

在科技行业中，潜伏的用户表征被广泛采用，以支持个性化的推荐系统。大多数先前的工作是用一个*单一的*高维嵌入来代表一个用户，这是一个很好的起点，但在提供对用户兴趣的全面理解方面还存在不足。在这项工作中，我们介绍了PinnerSage，一个端到端的推荐系统，它通过多模态的嵌入来代表每个用户，并利用这种丰富的用户代表来提供高质量的个性化推荐。PinnerSage通过将用户的行为聚类为概念上连贯的集群，并通过代表针（Medoids）对集群进行总结以提高效率和可解释性，从而实现这一目标。PinnerSage已经在Pinterest的生产中部署，我们概述了使其在非常大的范围内无缝运行的几个设计决定。我们进行了几个离线和在线A/B实验，表明我们的方法明显优于单一嵌入方法。

CCS概念

• **信息系统** → **推荐系统**；**个人化**；*聚类*；*近邻搜索*。

关键字

个性化推荐系统; 多模式用户嵌入

ACM参考格式：

Aditya Pal, Chantat Eksombatchai, Yitong Zhou, Bo Zhao, Charles Rosenberg, Jure Leskovec.2020.PinnerSage：多模式用户嵌入框架在Pinterest的推荐工作。In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery & Data Mining (KDD '20), Au- gust 23-27, 2020, Virtual Event, CA, USA.*ACM, New York, NY, USA, 10 pages. https://doi/10.1145/3394486.3403280

1 简介

Pinterest是一个内容发现平台，它允许3.5亿以上的月度活跃用户收集并与2B以上的视觉书签互动，这些书签被称为*针*。每个图钉都是一个与上下文相关的图片项目，代表了用户可以从世界各地找到并收藏的想法。用户可以把图钉保存在*板子上*，使它们有条不紊，易于查找。由于Pinterest上有数十亿个大头针，因此它成为了世界上最受欢迎的网站。

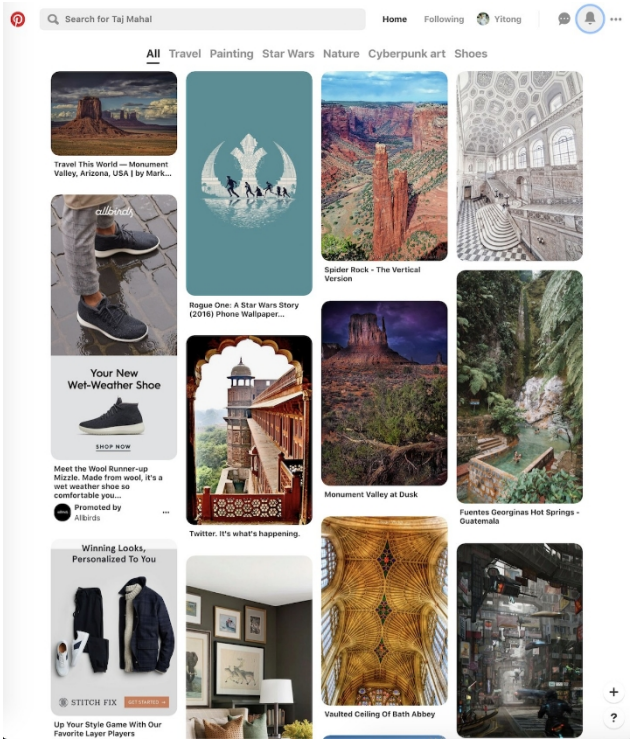

图1：Pinterest主页。

这对于帮助用户找到那些能激发灵感的想法（图钉）至关重要。因此，个性化的推荐构成了Pinterest用户体验的一个重要组成部分，并且在我们的产品中无处不在。Pinterest的推荐系统涵盖了各种算法，这些算法共同定义了主页上的体验。不同的算法针对不同的目标进行了优化，包括：（a）主页推荐，用户可以在主页上查看无限的推荐信息（如图1所示）；（b）购物产品推荐，链接到第三方电子商务网站；（c）个性化的搜索结果；（d）个性化的广告；（e）个性化的图钉板推荐，等等。因此，有必要对用户的兴趣进行普遍的、可共享的和丰富的理解，以便为Pinterest的大规模跨职能用例提供动力。

在促进我们对用户的理解方面，潜在的用户表示方法已经变得越来越重要。它们被证明能够有效地支持协同过滤技术[15, 20]，并作为排名模型的特征[6, 7, 26, 28]。由于它们的功效，用户嵌入被广泛采用在各种行业环境中。它们被用于支持YouTube和谷歌游戏

*这些作者贡献相同。

KDD '20，2020年8月23-27日，美国加利福尼亚州，虚拟活动。
Acm isbn 978-1-4503-7998-4/20/08.
https://doi.org/10.1145/3394486.3403280

推荐[6，7]，在Airbnb个性化搜索排名和类似房源推荐[11]，向用户推荐新闻文章[19]，在Etsy连接类似用户[30]，等等。

建立一个有效的用户嵌入系统，从数十亿的候选项目中向数以亿计的用户提供个性化的推荐，有几个固有的挑战。*首要的挑战是如何有效地对用户的多个方面进行编码。*

*一个用户？*一个用户可以有一系列不同的兴趣，但它们之间没有明显的联系。这些兴趣可以不断变化，有些兴趣是长期存在的，而有些兴趣则是在很短的时间内。大多数先前的工作旨在通过单一的高维嵌入向量来捕捉用户行为和兴趣的丰富多样性。通常情况下，要推荐的项目也在同一个嵌入空间中表示。这最初是令人满意的，但正如[3, 10, 16, 25]中描述的研究工作所指出的，一个好的嵌入必须编码用户的多种口味、兴趣、风格等，而一个项目（视频、图片、新闻文章、房屋列表、图钉等）通常只有一个焦点。因此，需要一个注意力层[29]或其他背景适应方法来跟踪用户不断变化的兴趣。

一个已经显示出前景的替代方案是用多个嵌入来表示一个用户，每个嵌入捕捉到用户的一个特殊方面。如[25]所示，多嵌入的用户表示可以在YouTube视频推荐中提供25%的改进。[16]也显示了小型基准数据集的合理收益。然而，由于先前的工作还没有完全解决几个重要的问题和担忧，多嵌入模型并没有在行业中广泛采用：

- 每个用户需要考虑多少个嵌入物？
- 如何对数以百万计的用户进行大规模推理并更新嵌入？
- 如何选择嵌入来生成个性化的建议？
- 在生产环境中，多嵌套模型是否会提供任何明显的收益？

大多数先前的多嵌入工作通过以下方式回避了这些挑战：要么只进行小规模的实验，不在生产中部署这些技术，要么将用户限制在很少的嵌入中，从而限制了这些方法的效用。

目前的工作。在本文中，我们介绍了一个端到端的系统，称为PinnerSage，已经部署在Pinterest的生产中。Pin- nerSage是一个高度可扩展的、灵活的和可延伸的推荐系统，它在内部用多个Pin- Sage[27]嵌入来代表每个用户。它通过对用户的行为进行分层聚类来推断多个嵌入，并通过medoids对聚类进行有效表示。然后，它采用了一个高效的近邻系统，为大规模推荐的候选者生成提供动力。最后，我们通过离线和在线实验对Pin- nerSage进行了广泛的评估。我们进行了几次大规模的A/B测试，表明基于PinnerSage的推荐为Pinterest的主页和购物产品推荐带来了显著的参与度。

2 销魂的设计选择

我们首先讨论了PinnerSage的重要设计选择。

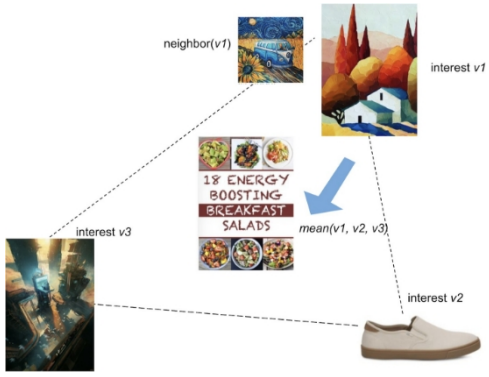

图2：256维嵌入的针脚在二维中被可视化。这些图钉描述了用户的三种不同兴趣。

设计选择1：针脚嵌入是固定的。大多数先前的工作是联合学习用户和物品嵌入（例如[7，11]）。这在大规模的应用中造成了固有的问题，例如它不必要地使模型复杂化，减慢了推理计算，并给实时更新带来困难。除此以外，我们认为它往往会导致不太理想的副作用。考虑到图2中的玩具例子，一个用户对*绘画*、*鞋子*和*科幻作品*感兴趣。联合学习用户和徽章嵌入会使这些不同主题的徽章嵌入更加接近，这实际上是我们希望*避免*的。针脚嵌入应该只在使相似的针脚更接近的基本原则下运作，同时尽可能地保持其余的针脚。为此，我们使用了PinSage[27]，它精确地实现了这一目标，没有任何稀释。PinSage是一个统一的针脚嵌入模型，它整合了视觉信号、文本注释和针脚板图形信息，以生成高质量的针脚嵌入。这种设计选择的另一个好处是，它大大简化了我们的下游系统和推理管道。

设计选择2：对嵌入的数量不加限制。以前的工作要么把嵌入的数量固定在一个小数目上[25]，要么给它们设定一个上限[16]。这种限制在最好的情况下会阻碍对用户的全面理解，在最坏的情况下会将不同的概念合并在一起，导致不好的建议。例如，合并项目嵌入，这被认为是合理的（见[7，28]），可能产生一个位于非常不同区域的嵌入。图2显示，合并三个不相干的品项嵌入的结果是一个嵌入，它最能代表概念*能量提升的早餐*。不用说，基于这种合并的推荐可能是有问题的。

我们的工作允许一个用户由其基础数据支持的尽可能多的嵌入来表示。这是通过分层聚类算法（Ward）将用户的行为聚类为概念上连贯的集群来实现的。一个轻度用户可能由3-5个聚类来代表，而一个重度用户可能由75-100个聚类来代表。

设计选择3：基于Medoids的集群表示。通常情况下，集群是由中心点表示的，这需要存储一个嵌入。此外，中心点可能对集群中的外来者敏感。为了紧凑地表示一个集群，我们挑选一个

簇成员针，称为medoid。根据定义，medoid是用户最初交互的pin集的成员，因此避免了主题漂移的隐患，并且对异常值有很好的适应性。从系统的角度来看，medoid是代表集群的一种简洁的方式，因为它只需要存储medoid的pin id，而且还可以导致跨用户甚至跨应用的缓存共享。

设计选择4：用于候选检索的Medoid抽样。PinnerSage通过集群的medoids提供了一个丰富的用户代表。然而，在实践中，由于成本问题，我们不能同时使用所有的medoids进行候选检索。此外，用户会被太多的不同项目所困扰。出于对成本的考虑，我们按照重要性分数（计算方法将在后面章节中描述）对3个中间物进行抽样，并推荐其最近的邻接针。媒介物的重要性分数每天都会更新，它们可以随着用户口味的变化而调整。

设计选择5：处理实时更新的双管齐下的方法。对于一个推荐系统来说，适应用户的当前需求是很重要的。同时，对用户的准确表述需要查看他们过去60-90天的活动情况。数据的巨大规模和增长速度使得我们很难同时考虑这两个方面。与[11]类似，我们解决了
通过结合两种方法来解决这个问题：(a)每天的批处理推理工作，根据每个用户的长期互动历史推断出多个medoids，以及(b)同一模型的在线版本，根据用户当天的互动情况推断出medoids。随着新活动的到来，只有在线版本被更新。在一天结束时，批处理版本会消耗当天的活动并解决任何不一致的地方。这种方法确保
我们的系统能迅速适应用户的当前需求，并在
同时又不损害他们的长期利益。

设计选择6：近似近邻系统。为了产生基于嵌入的推荐，我们采用了一个近似近邻（ANN）系统。给定一个查询（medoid），ANN系统在em-bedding空间中获取与查询最接近的k个针脚。我们展示了对ANN系统的一些改进，如过滤低质量的引脚，仔细选择不同的索引技术，缓存medoids；结果是最终生产版本的成本是原始原型的1/10。

3 我们的方法

符号。让集合P = {P_1, P_2, . .}代表Pinterest的所有图钉。P的cardinality是以亿为单位的。这里，$p_i \in R^d$表示第i个图钉的d*维*PinSage嵌入。让A_u = {a1, a2, . .}为用户u的行动图钉序列，这样，对于每个$a \in A_u$，用户要么*转发*，要么在时间$T_u[a]$*点击了*图钉P_a。为了简单起见，我们放弃下标u，因为我们为单个用户u制定，除非另有说明。我们认为A中的行动引脚是根据行动时间来排序的，这样，a_1是用户第一个行动的引脚ID。

主要假设：针脚嵌入是固定的。正如我们在设计选择1（第2节）中提到的，我们认为图钉嵌入是固定的，由一个黑盒模型生成。在Pinterest内部，这个模型是PinSage[27]，它被训练为在附近放置类似的图钉。

表1：预测下一个用户动作的模型的准确度提升。准确率是根据最后一个模型计算出来的。

型号	精度提升
最后一针	0%
衰减平均	25%
Kmeans Oracle	98%
甲骨文（*实际上限*）	**140%**

在嵌入空间中，以后续检索为目标。这个假设在我们的环境中是理想的，因为它大大简化了我们模型的复杂性。我们在之前的工作中也做了一个类似的假设[28]。

主要目标。我们的主要目标是推断出每个用户的多个嵌入，E = {e1, e2, . .}，其中$e_i \in R^d$为所有i，给定一个用户的行动A和引脚嵌入P。由于引脚嵌入是固定的，因此*不能*共同推断，我们寻求学习每个e_i与引脚嵌入兼容--*特别是为了检索与e_i相似的引脚*。对于不同的用户，嵌入的数量可以不同，即$|E_u|$*不*需要与$|E_v|$相同。然而，为了使我们的方法实际可行，我们*要求嵌入的数量在几十到几百之间*（|E| << |P|）。

为了展示基于聚类的方法的前景，我们考虑预测用户的下一个行动的任务。我们可以获得用户过去的行动{a1, a2, ...我们的目标是预测用户将与数十亿个图钉互动的下一个动作a_{i+1}。为了简化这一挑战，我们测量
通过问及 "你是怎么做到的？"来了解性能。
用户嵌入和引脚嵌入 P_a $_{i+1}$ 高于一个余弦值阈值为0
.8.我们比较了四种单一的嵌入方法：

(1) *最后的销路*：用户代表是最后一个行动销的嵌入（$_{Pai}$）。
(2) *衰减平均*：用户表征是他们行动销的嵌入的时间衰减平均数。具体来说，衰减平均嵌入 $\propto {}_{a \in A} e^{-\lambda(T_{now} - T[a])} - P_a$。
(3) *甲骨文公司*：甲骨文可以 "着眼于未来"，并作为用户挑选表示最接近a_{i+1}*的*用户过去的行动针。这就衡量了一个根据过去的约定来预测未来约定的方法的准确度上限。
(4) *Kmeans Oracle*：用户通过k-means聚类（k=3）代表他们过去的行动销。同样，神谕者可以看到针脚a_{i+1}，并挑选与之最接近的聚类中心点作为用户代表。

表1显示了结果。Oracle模型提供了可观的准确度提升，这是当之无愧的，因为它可以查看未来的引脚。然而，它的优越性能只是因为它能够回忆起所有针脚的嵌入（从系统的角度来看显然是不实际的）。有趣的是，基于聚类的Oracle只需要回忆3个聚类中心点的嵌入，就能比基线有很大的提高。这个结果并不完全令人惊讶，因为用户有多种兴趣，而且他们经常在这些兴趣之间转换。图3描述了这样一个例子，这在我们的环境中是非常多的。我们注意到，过去的5个引脚没有一个是相关的。

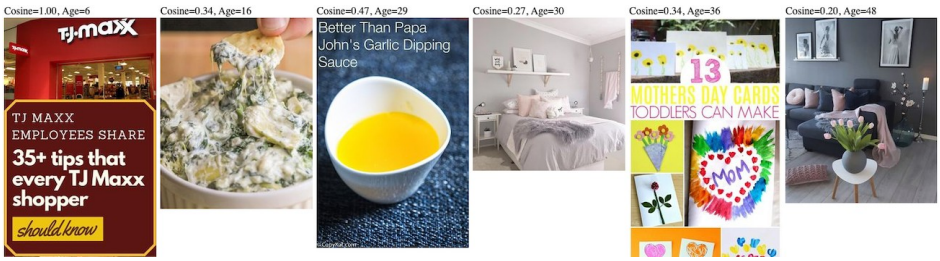

图3：一个随机用户的行动图钉（*重新插入*或*点击*）的快照。一个钉子的余弦分数是其嵌入和最新钉子之间的余弦相似度。销的年龄是指从行动日期到数据收集日期的天数。

与最新的引脚，人们不得不进一步寻找更强的相关性。因此，具有有限内存的单一嵌入模型在这个挑战中失败了。

3.1 萍水相逢

我们从前面的任务中得出两个关键的见解：(i)用单一的嵌入来代表一个用户是太有局限性了，(ii)基于聚类的方法可以在准确性和存储需求之间提供一个合理的权衡。这两个观点是我们的方法的基础，它有以下三个部分。

(1) 将用户在过去90天内的行动钉子拿出来，并将它们归类为少量的群组。
(2) 为每个集群计算一个基于medoid的表示。
(3) 计算每个集群对用户的重要性得分。

3.1.1 ***步骤1：对用户行为进行聚类。***我们对聚类方法的选择提出了两个主要的限制。

- 集群应该*只*结合概念上相似的针脚。
- 它应该自动确定集群的数量，以考虑到一个用户的不同兴趣数量。

Ward[24]满足上述两个约束条件，它是一种分层聚类方法，是基于最小方差标准的（满足我们的约束条件1）。此外，聚类的数量是根据距离自动确定的（满足我们的约束条件2）。在我们的实验中（第5节），它比K-means和完全联系方法[8]表现更好。其他几个基准测试也证实了Ward比其他聚类方法的性能更优越[1]。

我们对Ward的实现改编自Lance-Williams算法[14]，该算法提供了一种高效的聚类方式。

最初，算法1将每个引脚分配到它自己的群组。在每个后续步骤中，导致最小增幅的两个群组为

簇内方差被合并在一起。假设经过一些迭代，我们有了集群$\{C_1, C_2, ..\}$，集群C_i和C_j之间的距离表示为d_{ij}。那么如果两个聚类C_i和C_j被合并，距离将被更新如下：

$$d(C_i \cup C_j, C_k) = \frac{(n_i + n_k)\, d_{ik} + (n_j + n_k)\, d_{jk} - n_k\, d_{ij}}{n_i + n_j + n_k} \quad (1)$$

其中，$n_i = |C_i|$是群集i中的针脚数量。

沃德聚类算法的计算复杂度。沃德聚类的计算复杂度为$O(m^2)$，其中

[1] https://jmonlong.github.io/Hippocamplus/2018/02/13/tsne-and-clustering/

算法1：Ward(A = {a1, a2, ..}, α)

输入：A--用户的行动引脚
α - 集群合并阈值
输出：将输入引脚分组为群组
// 初始设置：每个针脚都属于自己的集群 设 $C_i \leftarrow \{i\}, \forall i \in A$
设 $d_{ij} \leftarrow \frac{1}{2}||P_i - P_j||^2 \; \forall i, j \in A$
merged_history = []
堆栈 = []
虽然|A|>1**做**
  //把A的第一个针脚（不去除）放到堆栈里。
  stack.add(A.*first*())
  while |*stack*| > 0 **do**
    $i \leftarrow$ *stack.top*()
    $J \leftarrow \{j : d_{ij} = \min_{i, j \in A} \{d_{ij}\}\}$.
    merged = *False*
    如果|*堆栈*|≥2**，那么**
      j = *stack.second_from_top*()
      如果$j \in J$**，那么**
        // 合并群组i和j
        stack.pop(); stack.pop(); //删除i和j
        merge_history.add($C_i \leftarrow C_j, d_{ij}$)
        $A \leftarrow A - \{j\}$
        设 $d_{ik} \leftarrow d(C_i \cup C_j, C_k), \forall k \in A, k \neq i$
        $C_i \leftarrow C_i \cup C_j$
        merged = *True*
    如果*merged* = *False*，**那么**
      //将*J*的第一个元素推入堆栈
      stack.push(*J*.*first*())
将*merged_history*中的图元按d_{ij}的递减顺序排序
设 $C \leftarrow \{\}$
置
foreach $(C_i \leftarrow C_j, d_{ij}) \in$ *merge_history* **do**
  如果$d_{ij} \leq \alpha$*并且*$\{C_i \cup C_j\} \cap C = \emptyset, \forall C \in C$**则**
    //将$C_i \cup C_j$添加到集群的集合中 Set C
    $\leftarrow C \cup \{C_i \cup C_j\}$
返回 C

$m = |A|^2$。为了看到这一点，我们注意到在每一个外部的while循环中，都有一个簇被添加到空栈中。现在，由于一个簇不能被添加到堆栈中两次（见附录，Lemma 8.2），该算法必须在循环过所有的*米数*后开始合并簇。

算法2： PinnerSage (A, α , λ) Set C

$\leftarrow$ Ward(A, α)
foreach $C \in C$ **do**
  设置 $medoid_C \leftarrow \arg\min_{m \in C} \sum_{j \in C} \|p_m - p_j\|^2$
  设 $重要性_C \leftarrow \sum_{i \in C} e^{-\lambda(T_{now} - T[i])}$
返回 $\{medoid_C, importance_C : C \in C\}$

集群（最坏的情况）。导致在堆栈中增加一个集群或合并两个集群的步骤，其计算成本为
$O(m)$。该算法先操作m*个初始集群*，然后随着进程操作m - 1个*中间合并集群*，导致总计算复杂度为$O(m^2)$。

3.1.2 ***第2步：基于Medoid的群集表示。*** 当一组针脚被分配到一个集群后，我们要为该集群寻找一个紧凑的代表。一个典型的方法是考虑集群中心点、时间衰减平均模型或更复杂的序列模型，如LSTM、GRU等。然而，上述技术的一个问题是，由它们推断出的嵌入可能位于d维空间中一个非常不同的区域。如果有离群的针脚被分配到集群中，这尤其是真的，这可能导致集群内的大差异。如图2所示，这种嵌入的副作用是检索出不相关的候选推荐。

我们选择了一种更稳健的技术，即选择一个称为*medoid的*集群成员针来代表该集群。我们选择与其他聚类成员的平方距离之和最小的那个针。与其他复杂模型得到的中心点或嵌入不同，medoid的定义是d维空间中与集群成员之一重合的一个点。从形式上看，

$$embedding(C) \leftarrow p_m \text{ ，其中} m = \arg\min_{m \in C} \sum_{j \in C} \|p_m - p_j\|^2 \quad (2)$$

medoid的另一个好处是，我们只需要存储medoid针的索引m，因为它的嵌入可以根据需要从一个辅助的键值存储中获取。

3.1.3 ***第三步：集群的重要性。*** 尽管一个用户的集群数量不多，但仍可能是几十个到几百个的数量。由于信息成本的原因，我们不能利用所有的集群来查询最近的邻居系统；因此必须确定集群对用户的相对重要性，这样我们就可以根据它们的重要性得分对集群进行抽样。我们考虑一个时间衰减平均模型
为了这个目的：

$$重要性(C, \lambda) = \sum_{i \in C} e^{-\lambda(T_{now} - T[i])} \quad (3)$$

其中，T[i]是用户对针脚i的行动时间。一个经常和最近被互动的集群将比其他集群有更高的重要性。设置λ=0会更加强调用户的频繁兴趣，而λ=0.1则会更加强调用户的近期兴趣。我们发现λ=0.01是这两方面的良好平衡。

算法2提供了一个PinnerSage模型的端到端概述。我们注意到，我们的模型对每个用户都是独立运行的，因此它可以*在*基于MapReduce的框架上相当*有效地并行*实现。我们还维护了一个在线

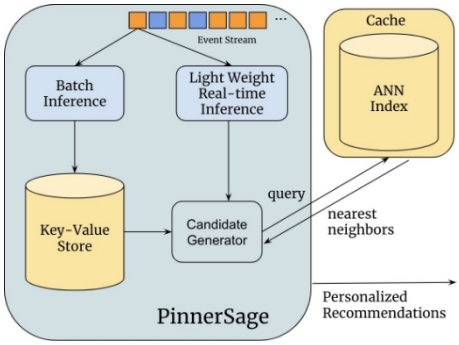

图4：PinnerSage推荐系统。

PinnerSage是在用户最近的活动中运行的版本。批量版本和在线版本的输出被结合在一起，用于生成推荐。

4 pinnersage推荐系统

PinnerSage可以为一个用户推断出基础数据所支持的尽可能多的medoids。从用户代表的角度来看，这是很好的，但是在任何时候都不能同时使用所有的medoids来生成推荐。为此，我们考虑了重要性抽样。在任何时候，我们对每个用户最多抽取3个medoids。然后，被抽样的medoids被用来从我们的近邻系统中检索候选人。图4提供了PinnerSage推荐系统的概况。

4.1 近似近邻检索

在Pinterest内部，我们有一个近似的最近的邻居重新三维系统(ANN)保持了一个有效的针孔索引。我们可以通过床单来检索与输入查询相似的针脚。

嵌入。由于它可以索引数十亿个引脚，因此确保其内部基础设施的成本和延迟在内部规定的限度内是一个工程挑战。我们讨论了一些技巧，这些技巧帮助ANN成为与其他候选检索框架（如Pixie[9]）一起的一流公民。

4.1.1 ***索引方案。*** 许多不同的嵌入索引方案（见[12]）被评估，如LSH Orthoplex [1, 21], Product Quantization [2], HNSW [17]等等。我们发现HNSW是在成本、延迟和召回方面表现最好。表2显示，HNSW

与LSH Orthoplex相比，HNSW的成本大大降低。在其他几个基准数据集上，HNSW的卓越性能也得到了报道[2].

候选库的细化。 对几十亿个针脚的完整索引会导致检索到许多近乎重复的东西。从推荐的目的来看，这些近似的重复是不可取的，因为将它们呈现给用户的价值有限。此外，一些图钉由于其美观性（低分辨率或图像中的大量文字）而具有内在的低质量。我们通过专门的内部模型过滤掉接近重复的和低质量的图钉。

[2] https://erikbern.com/2018/06/17/new-approximate-nearest-neighbor-benchmarks.html

表2：优化技术的相对成本效益。

优化技术	费
LSH Orthoplex → HNSW-60%	全
指数 → 指数	细化-50%
	集群中心

表2显示，指数细化导致了服务成本的显著降低。

缓存框架。对ANN系统的所有查询都是在针状嵌入空间中形成的。这些嵌入被表示为*d个*浮点值的数组，不适合缓存。另一方面，medoid的pin id很容易缓存，可以减少对ANN系统的重复调用。这对于作为medoids出现的多个用户的流行针脚来说尤其如此。表2显示了使用medoids比使用 centroids的成本降低。

4.2 服务模式

PinnerSage的主要目标是根据用户过去的参与历史向他们推荐相关内容。同时，我们希望提供与用户实时执行的行动相关的推荐。一种方法是将所有的用户数据反馈给PinnerSage，并在用户采取某种行动时立即运行。然而，由于成本和延迟问题，这实际上是不可行的：我们考虑一个双管齐下的方法：

(1) *每日批量推理*：PinnerSage每天在MapReduce集群上运行一个用户过去90天的行为。每日推理工作的输出（medoids列表和它们的重要性）在键值存储中在线提供。
(2) *轻量级的在线推理*：我们收集每个用户最近一天的20个行动（在最后一次更新键值存储中的条目之后）进行在线推理。PinnerSage使用一个实时的基于事件的流媒体服务来消费行动事件，并更新从键值存储发起的集群。

在实践中，系统优化在促成PinnerSage的生产中发挥了关键作用。表2显示了在实施过程中观察到的成本降低的粗略估计。虽然PinnerSage框架中存在某些限制，如双管齐下的更新策略，但该架构允许更容易地独立改进每个组件。

5 实验

在这里，我们对PinnerSage进行了评估，并从经验上验证了其性能。我们首先对PinnerSage进行定性评估，然后进行A/B实验，最后进行广泛的内部评估。

5.1 PinnerSage视觉化

图5是PinnerSage对一个特定用户的集群的可视化显示。可以看出，PinnerSage在生成概念上一致的集群方面做得很好，它只把上下文相似的图钉归类在一起。图6提供了一个由PinnerSage检索的候选人的图示。该推荐集是一个健康的组合，与用户的前三个兴趣相关的图钉：*鞋子*、*小玩意*、

表3：在Pinterest表面进行A/B实验。PinnerSage与当前生产系统的参与度。

实验量倾向		
家政服务	+4%	+2%
购物	+20%	+8%

和*食物*。由于该用户过去曾与这些主题进行过互动，他们很可能会发现这个多样化的推荐集很有趣，很有意义。

5.2 大规模的A/B实验

我们进行了大规模的A/B实验，用户被随机地分配到控制组或实验组。被分配到实验组的用户体验PinnerSage的推荐，而控制组的用户则从单一的嵌入模型（行动针的衰减平均嵌入）获得推荐。两组中的用户显示的推荐数量相等。表3显示，PinnerSage在增加整体参与量（转发和点击）以及增加参与倾向（每个用户的转发和点击）方面提供了显著的参与收益。任何收益都可以直接归因于PinnerSage推荐的质量和多样性的增加。

5.3 离线实验

我们进行了广泛的线下实验来评估PinnerSage及其变体与基线方法的关系。我们对大量的用户（*数以千万计*）进行采样，并收集他们在过去90天内的活动（*行动*和*印象*）。所有在*d*日之前的活动都被标记为训练，从*d日*开始的活动被标记为测试。

基线。我们将PinnerSage与下列基线进行比较：
(a) 单一的嵌入模型，如最后的针，衰减avg，有几个选择的λ（0，0.01，0.1，0.25），LSTM，GRU和HierTCN[28]，(b) PinnerSage的多嵌入变体，有不同的选择（i）聚类算法，（ii）集群嵌入计算方法，以及（iii）集群重要性选择的参数λ。

与[28]类似，基线模型的训练目标是用几种损失函数（l2、铰链、对数损失等）将用户行动排在印象之上。此外，除了印象，我们还用几种类型的否定词来训练基线，如随机图钉、流行图钉，以及通过选择与行动图钉相似的图钉来训练硬否定词。

评估方法。由一个模型为一个给定的用户推断出的嵌入对该用户的未来行为进行评估。测试批次按时间顺序处理，首先是*d*天，然后是*d*+1天，以此类推。一旦完成对一个测试批次的评估，该批次就被用来更新模型；模仿每日批次更新策略。

5.3.1 ***候选人检索任务的结果。*** 我们对用户嵌入的主要使用情况是，从一个非常大的候选池（数十亿个引脚）中检索出相关的候选者进行推荐。候选人检索集的生成方式如下：假设一个模型为一个用户输出了*e个*嵌入，那么$\lfloor \frac{400}{e} \rfloor$个最近的邻接插销检索每个嵌入物，最后，检索到的针脚将被用来作为一个整体。结合起来，形成一个大小为≤400的推荐集（由于重叠的原因

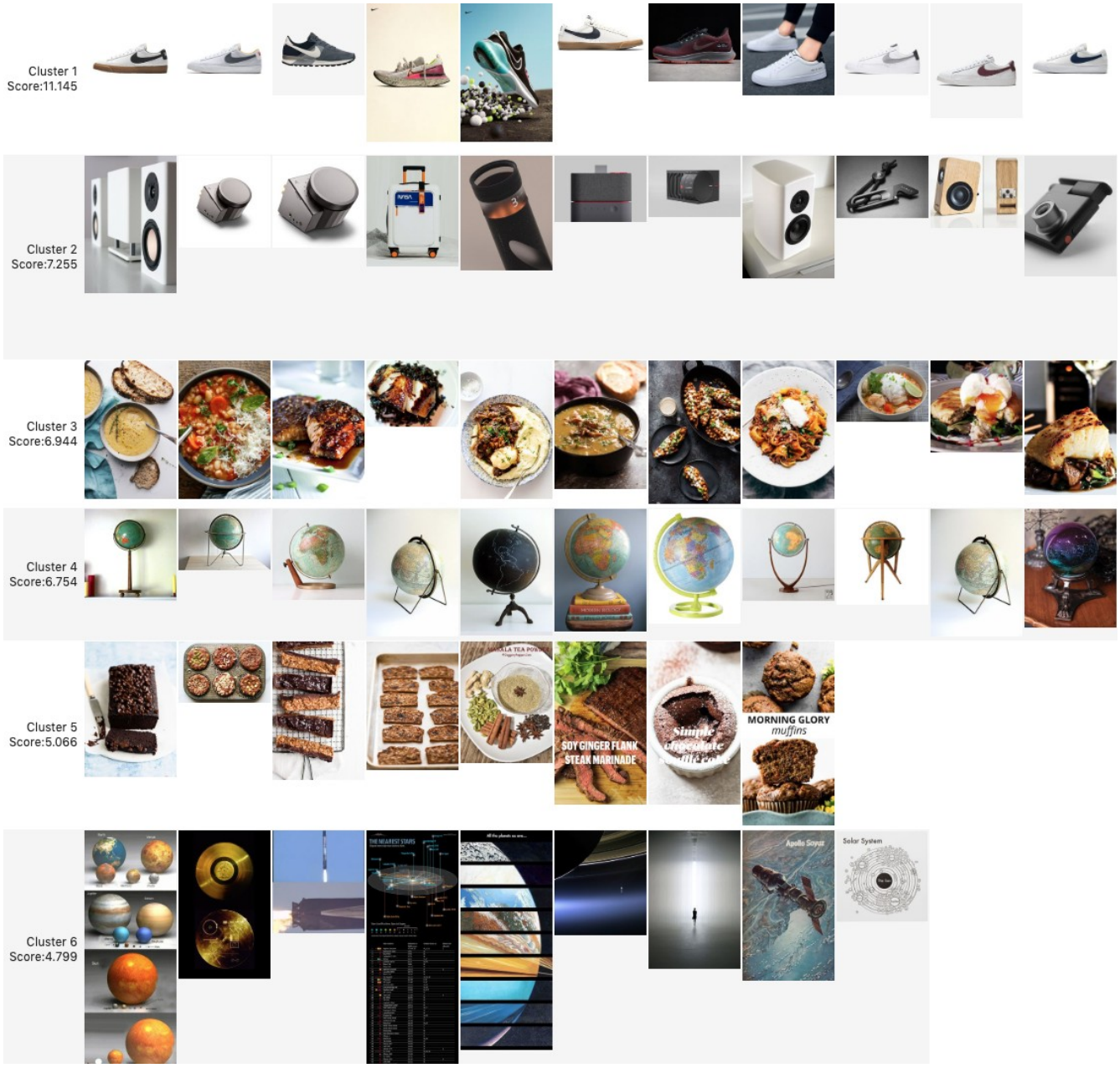

图5：一个匿名用户的PinnerSage集群。

图6：PinnerSage为前3个群组生成的建议样本 5.

表4：相对于检索任务的*最后一针模型的*提升。

	继电器。	*召回*
最后一针模型	0%	0%
衰减平均模型（λ=0.01）序	28%	14%
列模型（HierTCN）。	31%	16%
PinnerSage (样本1嵌入)	33%	18%
PinnerSage (K-means(k=5))	91%	68%
PinnerSage (Complete Linkage)	88%	65%
PinnerSage (embedding = Centroid)	105%	81%
PinnerSage (embedding = HierTCN)	**110%**	**88%**
PinnerSage (importance λ = 0)	97%	72%
PinnerSage (importation λ = 0.1)	94%	69%
PinnerSage (Ward, Medoid, λ = 0.01)	**110%**	**88%**

它可以小于400）。推荐集与测试批中观察到的用户行为一起在以下两个指标上进行评估：

(1) **相关性(*Rel.*)**是指观察到的与任何推荐针脚具有高余弦相似度（≥0.8）的行动针脚的比例。更高的相关性值会增加用户发现推荐集有用的机会。

(2) ***召回率***是指在推荐集中发现的行动销路的比例。

表4显示，PinnerSage在检索所有基线的相关候选人方面更加有效。特别是，PinnerSage的单一嵌入版本比最先进的单一嵌入序列方法更好。

在PinnerSage的变体中，我们注意到Ward的表现比K-means和完全链接方法更好。对于集群嵌入的计算，序列模型和medoid选择都有类似的表现，因此我们选择了medoid，因为它更容易存储，有更好的缓存特性。λ=0的集群重要性，与计算集群中的针数相同，比λ=0.01（我们的选择）表现得更差。从直觉上讲，这是有道理的，因为较高的λ值包含了与频率有关的经常性。然而，如果λ太高，那么它就会过度强调最近的兴趣，这可能会损害长期兴趣，导致模型性能下降（λ = 0.1 vs λ = 0.01）。

5.3.2 ***候选人排名任务的结果。***用户嵌入经常被用作排名模型中的一个特征，特别是对候选图钉进行排名。候选集是由测试批中的动作和印象针组成的。为了确保每个测试批的权重相同，我们对每个行动随机抽取20个印象。在一个给定的测试批中少于20个印象的情况下，我们增加随机样本以保持行动和印象的1：20比例。最后，根据与任何用户嵌入的最大余弦相似度的递减顺序，对候选引脚进行排名。一个更好的嵌入模型应该能够将行动排在印象之上。这种直觉是通过以下两个指标来体现的：

(1) **R-精度（*R-Prec.*）**是指在前k名中的行动销的比例，其中k是考虑对印象进行排名的行动数量。RP是衡量前k个排名项目中的信噪比。

表5：相对于排名任务的*最后一针模型的*提升。

	R-Prec.	*记录。级别*
最后一针模型	0%	0%
衰减平均模型（λ=0.01 序	8%	7%
列模型（HierTCN）	21%	16%
PinnerSage (样本1嵌入)	24%	18%
PinnerSage (Kmeans(k=5))	32%	24%
PinnerSage (Complete Linkage)	29%	22%
PinnerSage (embedding = Centroid)	**37%**	**28%**
PinnerSage (embedding = HierTCN)	**37%**	**28%**
PinnerSage (importance λ = 0)	31%	24%
PinnerSage (importation λ = 0.1)	30%	24%
PinnerSage (Ward, Medoid, λ = 0.01)	**37%**	**28%**

(2) **互惠排名（*Rec.Rank*）**是指行动图钉的平均互惠排名。它衡量的是行动销的排名有多高。

表5显示，PinnerSage的表现明显优于基线，表明它作为一个独立的特征产生的用户嵌入的功效。关于单一嵌入模型，我们的观察结果与检索任务相似：单一嵌入的PinnerSage推断出了更好的嵌入。在PinnerSage的变体中，我们注意到排名任务对嵌入计算不太敏感，因此中心点、medoid和序列模型有类似的表现，因为嵌入只用于对引脚排序。然而，它对集群的重要性分数很敏感，因为这决定了哪3个用户的嵌入被挑选出来进行排名。

5.3.3 ***多样化相关性的权衡。***推荐系统必须在相关性和多样性之间进行交易[5]。这对于重点有限的单一嵌入模型来说尤其如此。另一方面，多嵌入模型提供了同时覆盖不同的用户兴趣的灵活性。我们将多样性定义为推荐集之间的平均配对*余弦距离*。图7显示了多样性/相关性的提升，即最后的嵌入模型。我们注意到，通过增加e，我们同时增加了相关性和多样性。这在直觉上是有道理的，因为对于较大的e，推荐集是由跨越用户多种兴趣的相关图钉组成的。

对于e>3，相关度的提高会随着用户活动的增加而逐渐减弱。

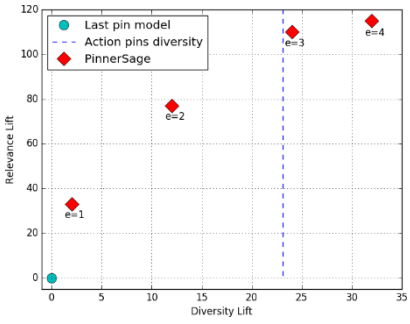

图7：选择不同数量的嵌入（e）进行候选检索时的多样性相关性权衡。

在给定的一天内（平均）不会有很大的变化。事实上，对于e=3，PinnerSage实现的推荐多样性与行动销的多样性密切匹配，我们认为这是一个甜蜜点。

6 相关的工作

有大量的研究工作集中在为用户和项目学习em- beddings，例如，[6, 7, 23, 26, 28]。这些工作的大部分是由提出的学习词代表的模型推动的，例如Word2Vec模型[18]，它是一个高度可扩展的连续词包（CBOW）和跳格（SG）语言模型。来自多个领域的研究人员已经在一些问题上采用了词代表学习模型，例如在不同的环境下进行推荐候选人排名，例如电影、音乐、工作、销路推荐[4, 13, 22, 28]。最近有一些论文也关注于候选人检索。

[6]提到候选人检索可以通过机器学习的模型和人类定义的规则的组合来处理；[23]考虑了从数十亿用户和项目中大规模地生成候选人，并提出了一个解决方案，为每个嵌入offiine预先计算出数百个类似项目。[7]讨论了一个基于单个用户嵌入的生产中的候选人生成和检索系统。

之前的一些工作[3, 16, 25]考虑用多个嵌入物对用户进行建模。[3]使用多个对时间敏感的背景资料来捕捉用户不断变化的兴趣。[25]考虑了因式分解模型中基于最大函数的非线性，等同于使用多个向量来代表一个用户，并显示了YouTube推荐中25%的改进。[16]使用多义嵌入（具有多种含义的嵌入）来改善节点表示，但它依赖于对每个嵌入的发生概率的估计来进行推理。这两篇文章[16, 25]都报告了对离线评估数据集的结果。我们的工作补充了之前的工作，并在此基础上展示了在生产环境中操作一个丰富的多嵌入模型。

7 结论

在这项工作中，我们提出了一个端到端的系统，名为PinnerSage，为Pinterest的个性化推荐提供动力。与之前基于单一嵌入的用户表征的生产系统相比，PinnerSage提出了一个基于多嵌入的用户表征方案。我们提出的聚类方案确保我们能够充分洞察用户的需求，并更好地了解他们。为了实现这一点，我们采用了几个设计选择，使我们的系统能够有效地运行，如基于medoid的集群表示和medoids的重要性采样。我们的内部实验表明，我们的方法为*检索任务*带来了显著的相关性收益，同时也为*排名任务*带来了*互惠排名*的改善。我们的大型A/B测试表明，PinnerSage提供了显著的现实世界的在线收益。我们的模型所带来的大部分改进可以归功于它对用户兴趣的更好理解和对他们需求的快速反应。在未来的工作中，我们考虑了几个有前景的领域，例如每个集群选择多个中间物，以及一个更系统的基于奖励的框架，在估计集群重要性时纳入隐性反馈。

8 鸣谢

我们要感谢Homefeed和Shop-ping团队在建立在线A/B实验方面的帮助。我们特别感谢嵌入基础设施团队为嵌入近邻搜索提供支持。

参考文献

[1] V.Athitsos, M. Potamias, P. Papapetrou, and G. Kollios.使用基于距离的散列法的最近邻居检索 。In *ICDE*, 2008.
[2] A.Babenko and V. Lempitsky.深度 描述符的十亿规模数据集的高效索引。In *CVPR*, 2016.
[3] L.Baltrunas and X. Amatriain.实现基于时间的推荐 隐性反馈。In *Workshop on context-aware recommender systems*, 2009.
[4] O.Barkan and N. Koenigstein.ITEM2VEC：用于协作的神经项目嵌入 tive filtering。In *Workshop on Machine Learning for Signal Processing*, 2016.
[5] J.G. Carbonell和J. Goldstein.使用mmr，基于多样性的重新排序，用于 重新排序文件和制作摘要。In *SIGIR*, 1998.
[6] H.Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson、G. Corrado, W. Chai, M. Ispir, R. Anil, Z. Haque, L. Hong, V. Jain, X. Liu, 和H.Shah.用于推荐系统的广泛和深度学习。In *DLRS@RecSys*, 2016.
[7] P.Covington, J. Adams, and E. Sargin.用于youtube推荐的深度神经网络-mendations.In *RecSys*, pages 191-198, 2016.
[8] D.Defays.一个完整链接方法的有效算法.*The Computer Journal*, 20(4):364-366, 01 1977.
[9] C.Eksombatchai, P. Jindal, J. Z. Liu, Y. Liu, R. Sharma, C. Sugnet, M. Ulrich, and J.莱斯科维茨Pixie：一个向2亿多 用户实时推荐30多亿个项目的系统。In *WWW*, 2018.
[10] A.Epasto and B. Perozzi.Is a single embedding enough? learning node represen-tations that capture multiple social context.In *WWW*, 2019.
[11] M.Grbovic and H. Cheng.在airbnb使用嵌入进行搜索的实时个性化 排名。In *KDD*, 2018.
[12] J.Johnson, M. Douze, and H. Jégou.Billion-scale similarity search with gpus. *CoRR*, abs/1702.08734, 2017.
[13] K.Kenthapadi, B. Le, and G. Venkataraman.在linkedin的个性化工作推荐 系统：实际挑战和经验教训。In *RecSys*, 2017.
[14] G. N. Lance and W. T. Williams.分类排序的一般理论 策略 1. 分层系统.*计算机杂志*，9（4）：373-380，1967年2月。
[15] G. Linden, B. Smith, and J. York.Amazon.com推荐：项目对项目 协作过滤。*IEEE互联网计算*，7（1）：76-80，2003。
[16] N.Liu, Q. Tan, Y. Li, H. Yang, J. Zhou, and X. Hu.单一矢量是否足够？探索网络嵌入的节点多义性。In *KDD*, pages 932-940, 2019.
[17] Y.A. Malkov和D. A. Yashunin.高效和稳健的近似最近 邻居搜索，使用分层的可导航小世界图。*PAMI*, 2018.
[18] T.Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean.分布式代表 词和短语及其构成性。在*NIPS*, 2013.
[19] S.Okura, Y. Tagami, S. Ono, and A. Tajima.基于嵌入的新闻推荐 dation for millions of users.In *KDD*, 2017.
[20] B.Sarwar, G. Karypis, J. Konstan, and J. Riedl.基于项目的协同过滤 推荐算法。In *WWW*, page 285âĂŞ295, 2001.
[21] K.Terasawa and Y. Tanaka.单位超球上的近似近邻搜索的球形LSH。In *Workshop on Algorithms and Data Structures*, 2007.
[22] D.Wang, S. Deng, X. Zhang, and G. Xu.学习嵌入音乐和元数据以实现情境感知的音乐推荐。*World Wide Web*, 21(5):1399-1423, 2018.
[23] J.Wang, P. Huang, H. Zhao, Z. Zhang, B. Zhao, and D. L. Lee.用于阿里巴巴电子商务推荐的亿级商品嵌入。In *KDD*, 2018.
[24] J.J. H. Ward.分层分组以优化目标函数.1963.
[25] J.Weston, R. J. Weiss, and H. Yee.通过嵌入 多个用户兴趣的非线性潜在因子。In *RecSys*, pages 65-68, 2013.
[26] L.Y. Wu, A. Fisch, S. Chopra, K. Adams, A. Bordes, and J. Weston.星空间：嵌入所有的东西!In *AAAI*, 2018.
[27] R.Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec.用于网络规模推荐系统的图卷积神经网络。In *KDD*, 2018.
[28] J.You, Y. Wang, A. Pal, P. Eksombatchai, C. Rosenberg, and J. Leskovec.Hierar-chical temporal convolutional networks for dynamic recommender systems.在*WWW*, 2019.
[29] N.Zhang, S. Deng, Z. Sun, X. Chen, W. Zhang, and H. Chen.基于注意力的胶囊网络与动态路由的关系提取。*arXiv预印本 arXiv:1812.11321*, 2018.
[30] X.Zhao, R. Louca, D. Hu, and L. Hong.为用户推荐学习项目-互动嵌入。2018.

可重复性补充材料

附录A：Ward 聚类算法的收敛性证明

定理8.1。*在Algo.1中，一个合并的簇$C_i \cup C_j$与另一个簇C_k的距离不能低于其子簇与该簇的最低距离，即*$d(C_i \cup C_j, C_k) \geq min\{d_{ik}, d_{jk}\}$。

证明。对于集群i和j的合并，必须满足以下两个条件：$d_{ij} \leq d_{ik}$和$d_{ij} \leq d_{jk}$。在不丧失一般性的情况下，让$d_{ij} = d$，$d_{ik} = d + \gamma$，$d_{jk} = d + \gamma + \delta$，其中$\gamma \geq 0$，$\delta \geq 0$。我们可以将公式1简化如下：

$$d(C_i \cup C_j, C_k) = \frac{(n_i + n_k)(d + \gamma) + (n_j + n_k)(d + \gamma + \delta) - n_k d}{n_i + n_j + n_k}$$

$$= d + \gamma + \frac{n_k \gamma + (n_j + n_k)\delta}{n_i + n_j + n_k} \geq d + \gamma \quad (4)$$

$d(C_i \cup C_j, C_k) \geq d + \gamma$意味着$d(C_i \cup C_j, C_k) \geq min\{d_{ik}, d_{jk}\}$。 □

定理8.2。*在Ward聚类中，一个簇不能被两次添加到堆栈中（算法1）。*

矛盾的证明。让堆栈在某一特定时间的状态为[…，i，j，k，i]。由于j在堆栈中被加在i之后，这意味着$d_{ij} \leq d_{ik}$（条件1）。同样地，从堆栈的后续添加中，我们得到$d_{jk} \leq d_{ji}$（条件2）和$d_{ki} \leq d_{kj}$（条件3）。我们还注意到，通过对称性，$d_{xy} = d_{yx}$。结合条件2和条件3可以得出$d_{ik} \leq d_{ij}$，这将与条件1相矛盾，除非$d_{ji} = d_{jk}$。由于i是堆栈中加入j后的第二个元素，鉴于$d_{ji} = d_{jk}$，j不能加入k。因此，i不能被两次添加到栈中。

此外，自第一次将i添加到堆栈后的集群合并，不能再添加i。这是因为它与i的距离是
大于或等于它的子集群与其他集群的最小距离
i，根据定理8.1。由于最接近i的子群不能加入i，所以
不能合并的集群。 □