

Embedding在腾讯应用宝的推荐实践

原创 carloslin 腾讯技术工程 8月18日



作者：carloslin，腾讯 PCG 应用研究员

Embedding 技术目前在工业界以及学术界中应用非常广泛，关于 Embedding 的探索和应用从未停歇。Embedding 的训练方法主要分成 DNN 的端到端的方法以及序列学习的非端到端的方法，其中最经典的 word2vec 以及由此衍生出 sentence2vec, doc2vec, item2vec 等都属于非端到端的学习方法；本文主要介绍 Embedding 技术的非端到端学习方法在应用宝推荐场景的应用实践。

1.经典的 word2vec 模型

word2vec 模型的问世，极大的促进了 Embedding 技术的发展。下面我们先从 word2vec 模型切入，简单介绍一下 embedding 的推导过程。以业界最广泛使用的 Skip-gram+negative sampling 为例。

损失函数如下所示：

Loss function :

$$\operatorname{argmax}_{\theta} \sum_{(a,c) \in D_p} \log \frac{1}{1 + e^{-v'_c v_a}} + \sum_{(a,c) \in D_n} \log \frac{1}{1 + e^{v'_c v_a}}$$

↑正样本
 ↑全局随机负样本

其中 a 表示当前中心词，c 表示序列上下文词，D_p 为 window_size 中的词集合，D_n 为全局随机采样的负样本集合。损失函数中左半部分为正样本的损失函数项，右半部分为负样本的损失函数项。

为什么正样本和负样本能够通过加法的方式组合起来构成损失函数呢？ 首先，原始的 skip-gram 模型的损失函数 (1) 如下：

$$\operatorname{argmax}_{\theta} \prod_{(w,c) \in D} p(c|w; \theta)$$

其中 w 为中心词, c 为 w 的上下文单词, D 为训练集中 w 和 c 的 pair 对, θ 为需要学习的参数; 我们可以把这个优化问题转化为一个分类问题, 利用 softmax 函数展开后:

$$p(c|w; \theta)$$

我们得到如下格式的函数 (2) :

$$p(c|w; \theta) = \frac{e^{v_c \cdot v_w}}{\sum_{c' \in C} e^{v_{c'} \cdot v_w}}$$

其中 v_c 和 v_w 分别为中心词和上下文词的 embedding, C 为训练集中所有上下文单词的集合。接着我们可以对损失函数 (1) 进行 log 变换得到 (3) :

$$\arg \max_{\theta} \sum_{(w,c) \in D} \log p(c|w) = \sum_{(w,c) \in D} (\log e^{v_c \cdot v_w} - \log \sum_{c'} e^{v_{c'} \cdot v_w})$$

由于需要对所有的上下文单词集合进行计算, 公式(3)的计算复杂度非常高, 所以 Negative sampling 的训练方式应运而生。Negative sampling 的思想本质上是一个**二分类**的问题, 即预测 (w, c) pair 是否存在训练集中。我们用公式:

$$p(D = 1|w, c; \theta)$$

表示 (w, c) 存在训练集的概率, 相应的:

$$p(D = 0|w, c) = 1 - p(D = 1|w, c)$$

表示 (w, c) 不存在训练集中的概率。此时的我们的损失函数 (4) 如下所示:

$$\arg \max_{\theta} \prod_{(w,c) \in D} p(D = 1|w, c; \theta) \rightarrow \arg \max_{\theta} \sum_{(w,c) \in D} \log \frac{1}{1 + e^{-v_c \cdot v_w}}$$


即我们把实际出现过的 (w, c) pair 对当成了正样本, 损失函数的目标就是希望能学习到参数 θ 来最大化 (w, c) 作为正样本的概率, 通过对:

$$p(D = 1|w, c; \theta)$$

进行 sigmoid 函数和 log 转换, 我们得到了上式右半部分的公式, 如此便得到了我们正样本的损失函数表达。但是并不是所有 (w, c) pair 对都来自训练集合, 因此需要构造一些负样本来

修正损失函数拟合正样本的概率表达，最终 SGNE 的损失函数如下所示：

$$\arg \max_{\theta} \prod_{(w,c) \in D} p(D=1|c,w;\theta) \prod_{(w,c) \in D'} p(D=0|c,w;\theta)$$



$$\arg \max_{\theta} \sum_{(w,c) \in D} \log \frac{1}{1+e^{-v_c \cdot v_w}} + \sum_{(w,c) \in D'} \log \left(\frac{1}{1+e^{v_c \cdot v_w}} \right)$$

其中 D'即为全局随机采样的负样本。

2.应用宝相关推荐场景介绍

接下来我们来介绍一下 Embedding 技术在应用宝相关推荐场景中的应用实践。首先，简单介绍一下应用宝相关推荐场景的概况。应用宝相关推荐场景主要包括详情页、OMA（one more app）以及下载管理和应用更新。推荐的方式主要是根据上文 APP（详情页当前 APP、OMA 正在下载的 APP）召回相关 APP 进行推荐。



图1 详情页相关推荐场景 图2 OMA相关推荐场景

3.传统 word2vec 的不足

传统的 word2vec，在实际应用中存在一些不足：

- 只能学习训练数据中 window_size 内当前词和上下文词的相关性，无法表达未登录词与当前词的相关性；
- 负样本和正样本的定义无法表达上下文场景中上文和下文的关系。

应用宝的业务特点存在以下三个挑战：

- 从用户层面来看，用户月均下载 APP 量级仅个位数，用户行为非常稀疏；
- 从 APP 层面来看，APP 流量差异巨大，75%的 APP 下载集中在 top1000 个 APP，大量长尾 APP 行为稀疏，传统的序列建模无法准确学习长尾 APP 的 embedding 表达；
- 在相关推荐场景，看重上下文的相关性、相似性，从业务角度看，召回 APP 与上文 APP 需要在类目层面上有相关度，传统的序列建模无法表达这个信息。

接下来我们将会从用户序列的样本优化以及模型优化来解决上述三个挑战。

4.用户序列优化

4.1 长周期用户序列优化

线上 Base 流量的解决方案是通过拉长数据周期，扩充训练样本，选取过去 180 天的用户的下载行为序列作为训练数据。这种方法存在一个问题，下载序列的周期跨度过长，APP 下载行为间隔大，APP 的下载之间几乎没有相关性，同时在短期的下载序列中，是有一定的相关性的。以下图的下载序列为例子，上半部分为 10 月 25 号的下载行为，几乎都是交友类的 APP，而下半部分为 11 月 07 号的下载行为，几乎都是购物类 APP，两者并没有相关性。

用户ID	下载时间	下载APP	上文APP
1	20191025 22:43:30	野玫瑰	MOMO陌陌
	20191025 22:43:35	单身交友	同城热恋
	20191025 22:43:41	附近约爱	同城热恋
	20191025 22:43:43	缘多多	觅恋
	20191025 22:43:50	附近爱约会	附近约爱
	20191025 22:43:53	单身约爱	附近约爱
	20191120 12:06:59	招商银行掌上生活	招商银行
	20191120 12:07:00	中国建设银行	招商银行
	20191120 14:17:33	得物(毒)	京东
	20191229 12:42:37	U号租	vv租号
2	20191107 09:26:01	淘宝特价版	手机淘宝
	20191107 09:28:14	淘集集	淘宝特价版
	20191107 09:31:16	唯品会	京东
	20191107 09:31:30	一淘—淘宝官方返利、优惠券、	唯品会
	20191107 09:36:26	天猫	唯品会
	20191107 09:36:27	寺库奢侈品	唯品会
	20200127 16:28:59	内存清理	360清理大师
	20200127 16:29:05	腾讯手机管家—QQ微信保护	360清理大师
	20200207 21:18:50	猎豹安全大师	一键清理

图3 用户行为序列示意图

因此我们以天为 session 粒度，重新构建用户的下载行为序列。在序列长度的设定上，过短的序列，无法有效表达行为之间的相关性，不利于模型学习 APP 的 embedding 表达。在我们相关推荐场景中，设定的阈值为 5。模型训练方式采用传统的 Skip-gram+negative sampling 的方式训练。通过拉长数据周期的方式扩充训练样本，使得我们的训练样本数据增长了 5 倍。

4.2 引入图随机游走模型

在上一步的迭代中，虽然我们通过拉长数据周期的方式扩充训练样本，但是由于用户行为的稀疏性，天级别 session 粒度的下载序列样本非常少，仅占 5%，因此并不能很好的解决长尾 APP 学习困难的问题。

因此借鉴图游走算法的思想，我们利用用户的 APP 下载序列，构造用户的下载行为图，并在图中做随机游走，生成新的行为序列。

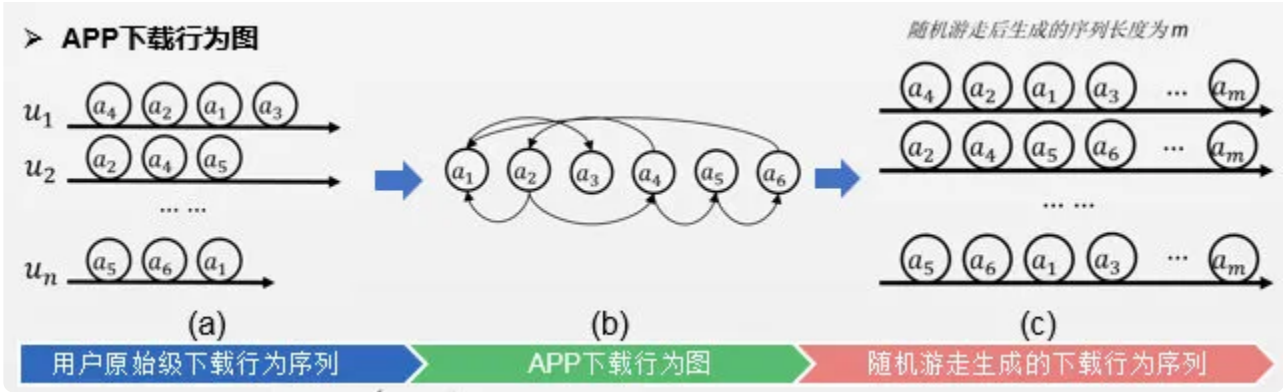


图4 随机游走示意图

如上图所示，我们有三个原始的用户行为序列 u1/u2/u3，其中 a 表示行为 APP。首先我们把序列中各个 APP 抽取出来构造图节点，序列的先后行为发生关系构造节点之间的有向边来构造 APP 的行为转换图（如图 4-(b)图所示），接着我们在图中进行随机游走，生成长度为 m 的新的行为序列作为我们的训练数据。

其中随机游走的概率公式如下：

$$P(v_j|v_i) = \begin{cases} \frac{M_{ij}^{dist}}{\sum_{j \in N_+(v_i)} M_{ij}^{dist}}, & v_j \in N_+(v_i) \\ 0, & e_{ij} \notin E \end{cases}$$

其中：
M 为行为的邻接矩阵
N₊(v_i) 为节点 v_i 的所有邻居节点
dist 为失真幂，用于对热门 APP 做转移降权

其中邻接矩阵以图 4-(a)图三个原始行为序列为例，邻接矩阵如下所示：

目标节点
a ₁ a ₂ a ₃ a ₄ a ₅ a ₆

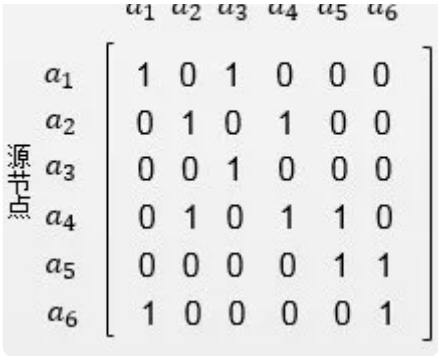


图5 用户行为序列邻接矩阵图

下图为 randomwalk 后 app 的样本分布图：

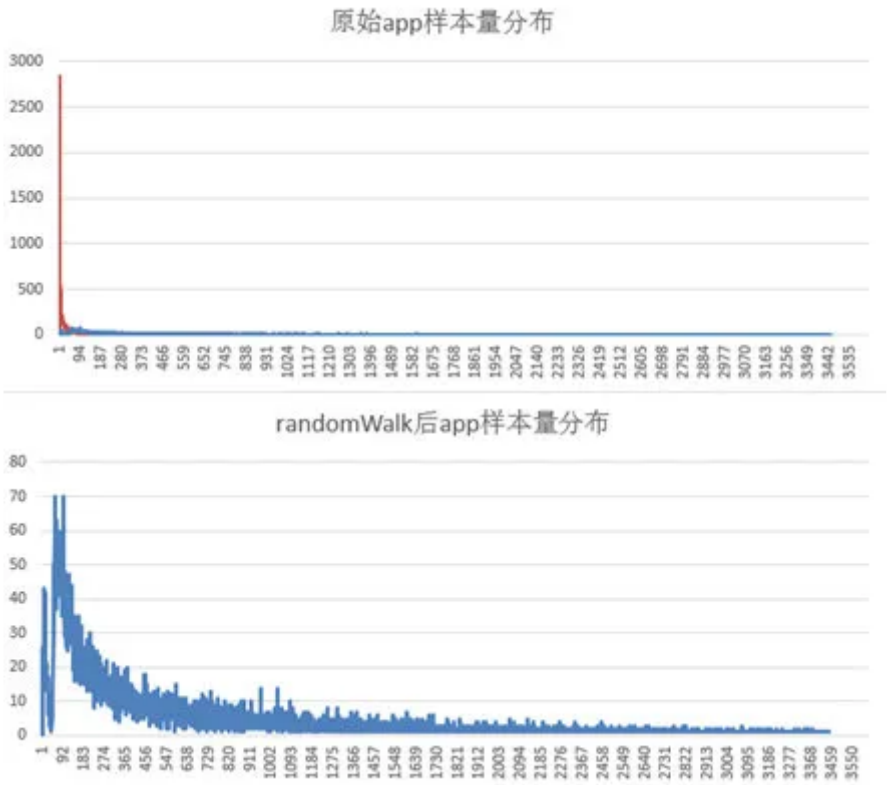


图7 APP样本分布图

app样本数	原始APP样本数占比	RandomWalk后样本数占比
1~10	35.43%	1.47%
10~100	27.62%	16.75%
100~1000	25.69%	53.05%
1000~10000	10.17%	23.44%
10000+	1.10%	5.28%

图8 APP样本分布占比变化图

从图 7 可以发现 randomwalk 能有效缓解长尾 APP 训练样本不足的问题，randomWalk 前后对比，APP 的样本分布相比更加均匀；图 8 展示样本数为 1~10 的 APP 占比从 35%下降到 1.47%，同时 APP 样本数在 100~10000 的占比从 35%左右提升到了 75%以上，低频长尾 APP 的样本得到增强。

在这一步优化中我们通过构造用户的 APP 下载行为序列图，并通过随机游走的方式生成新的用户行为，极大的提高了长尾 APP 的训练样本量，解决长尾 APP 学习不充分的问题。但是这里 APP 下载行为序列图的在随机游走的过程中并没有体现上文 APP 对下载行为的影响。

4.3 图游走+约束采样

在应用宝的相关推荐场景中，以详情页、OMA 场景为例：进入详情页有一个前置行为，即用户首先需要**点击上文 APP**，才有可能进入到详情页。OMA 场景则需要用户**点击下载上文 APP**，才会出现相关推荐卡片。

比如我们在首页中展示的 APP feeds 如下图，以第一个 APP “七猫免费小说”为例，当我们点击七猫免费小说后，我们会进入详情页，当我们点击下载按钮，会弹出 OMA（one more app）卡片。



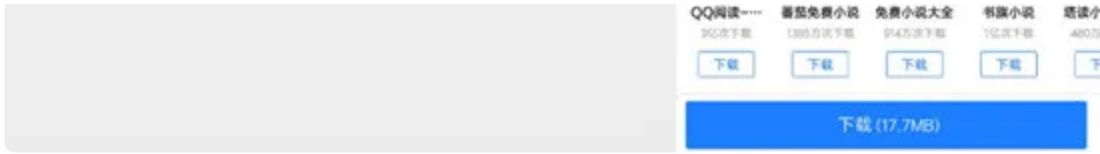


图9 详情页、OMA相关推荐逻辑示意图

这里都表达了用户的对上文也是感兴趣的，但是上文的信息在传统的 word2vec 或随机游走算法中并没有考虑。因此在这里我们通过对 randomWalk 的**引入上文 APP 的约束**，只有出现过同上文的 APP 会被采样，从训练样本的层面引入上文 APP 的信息。首先我们在随机游走概率公式中加入同上文的约束：

随机游走概率公式：

$$P(v_j|v_i) = \begin{cases} \frac{M_{ij}^{dist}}{\sum_{j \in N_+(v_i)} M_{ij}^{dist}}, & v_j \in N_+(v_i), R(v_i) \cap R(v_j) \neq \emptyset \\ 0, & e_{ij} \notin E \end{cases}$$

随机游走时加入同上文约束

其中下面两个变量分别表示节点 i 和 j 的上文集合：

$R(v_i)$

$R(v_j)$

游走过程的示意图如下：

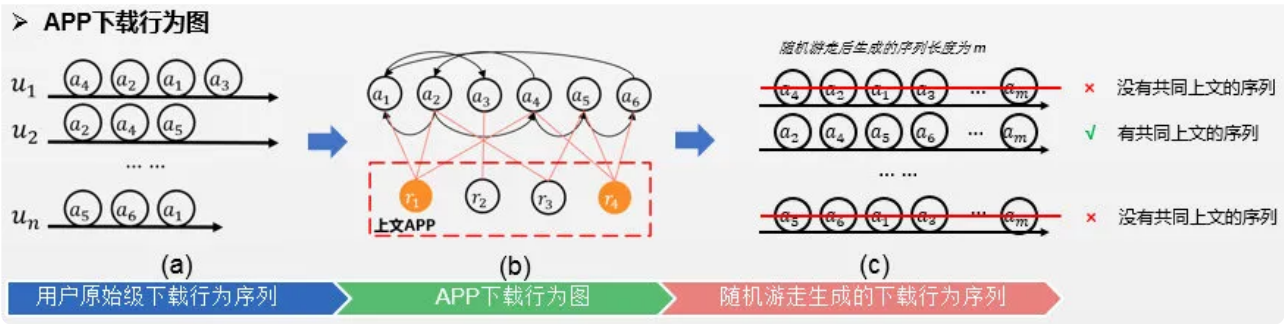


图10 引入上文约束的随机游走示意图

从图 9- (b) 中可以看到，下半部分我们引入了各个 APP 的上文 APP 信息，并且在 (c) 中，如 a4->a2->a1->a3->...->am 将不会出现在训练数据中，因为 a1 和 a3 没有共同的上文；a5->a6->a1->a3->...->am 也不会出现，因为 a1 和 a6 没有共同的上文。

通过引入上文约束的随机游走方式生成的样本，初步地表达了上文信息对于下载行为序列的影响；后续我们将尝试更多的图算法如 GraphSage 在我们场景应用实践。

比如我们对比可爱女生闹钟（长尾 APP）的召回推荐结果，相比传统的 i2v，i2v+randomWalk 的方式召回结果更加相似。

上文	i2v召回	i2v+randomWalk召回
可爱女生闹钟	美日记	睡你妹闹钟
	时光日记本	小日常
	爱笔记	正点闹钟
	浅言	萌萌闹钟
	一本日记	番茄ToDo
	堆糖	最美闹钟
	日记本	万能定时器
	可爱日历	准了
	i豆闹钟	小睡眠
	小陪伴	怪物闹钟

图11 i2v召回与i2v+randomWalk 长尾APP召回Case对比图

5.模型优化

上一 part 的优化中，我们主要是从样本层面对长尾 APP 的稀疏性以及上文的约束性进行优化，接下来我们从模型层面来优化模型的相关性。传统的 word2vec 模型的损失函数中只有 window_size 中的正样本以及全局随机采样的负样本，借鉴 air-bnb embedding 的思想，我们在损失函数中引入上文 APP 作为序列的全局 context，引入同类目随机采样负样本

在 air-bnb 的 paper *Real-time Personalization using Embeddings for Search Ranking at Airbnb* 中，通过引入 booked-listing 作为序列的全局 context，相当于对于这一条用户行为序列，把 booked-listing 作为正样本引入到损失函数中。

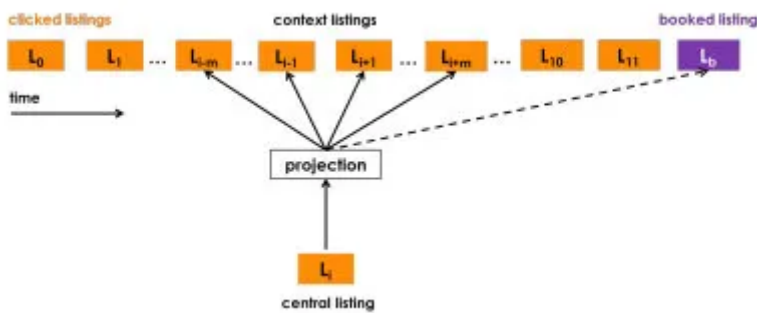


图12 AirBnb embedding序列图

5.1 正样本 Loss 优化

借鉴 airbnb-embedding 的思想，在相关推荐场景中，用户下载相关 APP 首先是基于对上文 APP 感兴趣才会进入详情页或展示 OMA 相关 APP，因此把上文 APP 信息引入到模型中一起学习是 make sense。

如下图是传统的 SGNE 的损失函数，正样本为 window_size 内的 app，负样本为全局随机采样的 APP。

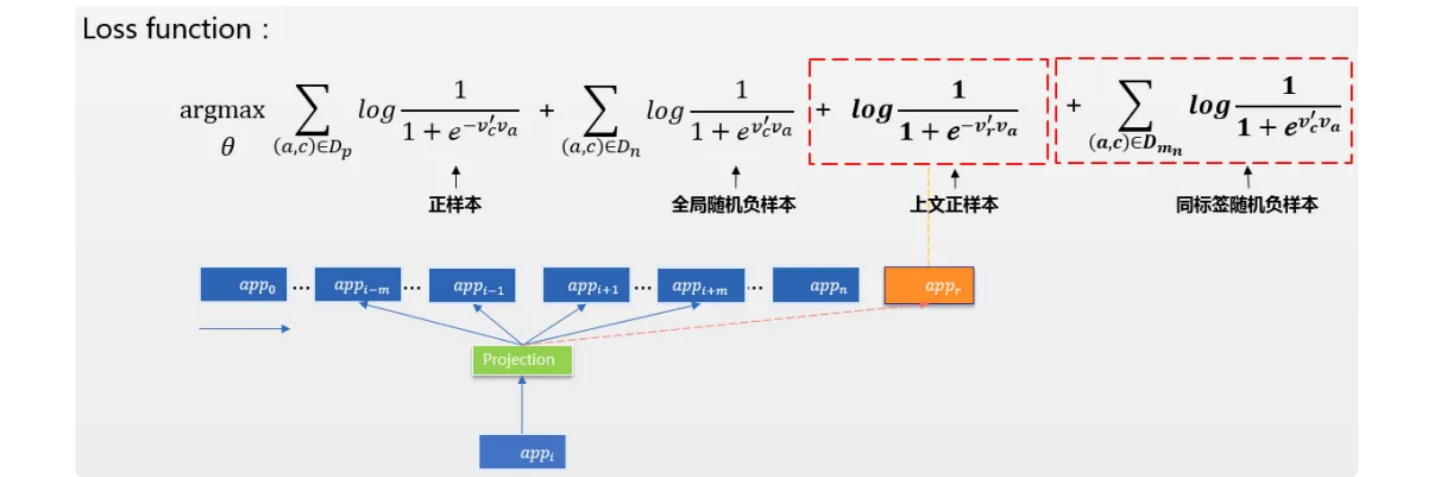
景中，我们的做法是，全局随机负采样的同时，对当前 APP 的同类目 APP 也进行随机负采样。

全局随机负采样得到的负样本可以理解为 easyexample，因为这部分负样本只有很小的概率是来自 window_size 中的(w,c)组合，模型很容易学习；而对于同类目采样的得到的负样本为 hard example，因为这部分样本本身是有一定的相关性的（同类目），我们希望让模型能够学习到同类目 APP 中的内部差异。具体地我们会从当前 app_i 的同一级类目中随机采样 APP 作为负样本。

其中

$$D_{m_n}$$

表示从 m 个同一级类目 app 中随机采样的负样本。



可爱女生闹钟	时光日记本	正点闹钟
	爱笔记	i豆闹钟
	浅言	睡你妹闹钟
	一本日记	目标倒计时
	堆糖	最美闹钟
	日记本	时间规划局
	可爱日历	准了
	i豆闹钟	准点闹钟
	小陪伴	怪物闹钟

图16 模型优化的case召回结果对比图

6.小结

为了解决用户和 APP 的行为稀疏性，我们首先通过拉长数据周期的方式扩充训练样本；然后利用 randWalk 的方式，有效提升长尾 APP 的样本比例，从一定程度上缓解了长尾 APP 的样本不足导致学习不充分的问题；在模型层面，我们通过引入上文的辅助 loss 使得模型能够学习到上下文 APP 的相关性，引入当前 APP 同类目的负样本学习同类目 APP 内部的差异性。

