基于用户行为分析的智能终端应用管理优化①

黄文茜[1,2], 李 梅[1,2], 于佳耕[1]

[1](中国科学院软件研究所 基础软件国家工程研究中心, 北京 100190)
[2](中国科学院大学, 北京 100190)

摘 要: 随着智能终端的普及, 涌现了多种多样的应用程序以满足用户需求. 现有智能终端系统普遍使用基于 LRU 算法的 Task killing 机制管理后台应用程序, LRU 算法只考虑了应用最近的使用情况, 没有考虑用户使用习惯, 可能导致后台应用程序被错误地终止, 当用户切换回该应用程序时, 会带来应用启动延迟增加、能耗增加、状态丢失等问题. 本文设计并实现了一种基于贝叶斯网络的应用管理方法 BNLP, 并在 Android 移动终端上验证. 该方法通过分析用户使用行为, 预测后台应用程序即将被启动的概率, 并据此进行应用管理. 在 LiveLab 数据集上的实验表明, 本文提出的 BNLP 模型相比于 LRU 算法应用程序重启率降低了 17.2%, 从而降低了延迟和能耗、提升了用户体验.

关键词: 用户行为分析; 智能终端; 应用管理; 贝叶斯网络

Application Management Optimization for Smart Terminals Based on User Behavior Analysis

HUANG Wen-Qian[1,2], LI Mei[1,2], YU Jia-Geng[1]

[1](Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)
[2](University of Chinese Academy of Science, Beijing 100190, China)

Abstract: With the widespread of smart terminals, there appear a lot of applications to satisfy all kinds of needs in users' daily lives. Existing systems of smart terminals manage the background applications through an LRU-based task killing policy. LRU algorithm only considers applications' recent using logs instead of users' application using habits, it'll increase the application restart ratio and also lead to problems like increasing application launch delay, increasing energy consumption and losing state. In this paper, it designs and implements a Bayesian network-based application management policy named BNLP, which analyzs users' application using behavior to predict the probability of launching each application in the near future. Then BNLP will manage the background applications according to these probabilities. Experiment results show that our BNLP model can decrease the restart ratio of LRU by 17.2%, which greatly reduces launch delay, energy consumption and improves user experience.

Key words: user behavior analysis; smart terminal; application management; Bayesian network

随着智能手机的逐渐普及, 涌现了大量的应用程序以满足用户生活中方方面面的需求. 而用户在使用手机的时候, 存在频繁的应用程序切换. 若每次切换都重新启动相应的应用程序, 需要重新创建进程、读文件等初始化操作, 会增加启动延迟、增加能耗. 并且重新启动会丢失用户之前的使用状态, 而迅速恢复到之前的使用状态对于好的用户体验至关重要. 综上所述, 为了降低延迟、快速恢复状态以提升用户体验, 降低能耗以提升系统性能, 减少应用程序的重新启动变得尤为重要.

针对这个问题, Android 系统维护了一定数量(MAX_HIDDEN_APPS)的后台进程, 当用户切换到这些后台进程的时候, 不需要重新启动进程, 省去了大量创建和初始化的开销, 可以迅速恢复到之前的使用

① 基金项目:国家自然科学基金(61432001,91218302,61402451)
收稿时间:2016-01-27;收到修改稿时间:2016-03-08 [doi:10.15888/j.cnki.csa.005359]

状态. 当后台进程数达到 MAX_HIDDEN_APPS 会触发基于 LRU 的 Task killing 机制, 终止最近最少使用的应用程序, 将其清理出内存. 然而基于 LRU 的方法只考虑了最近的应用程序使用情况, 并没有考虑用户的应用程序使用习惯, 可能终止了即将启动的应用程序, 从而造成较多的应用程序重启动. 除此之外, LRU 的效果很依赖于 MAX_HIDDEN_APPS 值的大小, MAX_HIDDEN_APPS 的值越大, 效果越好. 然而在内存中维护大量的后台进程本身也需要一些管理开销. 研究表明较大的 MAX_HIDDEN_APP 不仅会带来更多的内存管理开销, 而且可能带来内存泄露的风险[1]. 因此, 在 MAX_HIDDEN_APPS 较小的时候保持较低的应用程序重启率变得尤为重要.

本文提出了一种基于贝叶斯网络的应用管理方法, 综合利用了用户 App 使用日志、情景信息、App 间关联等丰富的信息, 这些信息对于准确预测用户行为至关重要. 例如, 用户在办公室可能会使用一些办公类软件, 在家的时候更多的使用社交游戏影音类软件, 而在户外的时候则可能使用地图类软件. 又例如, 用户在周末和在工作日使用软件的行为可能很不同. 另外, App 间本身具有一些关联. 例如, 用户喜欢在浏览网页的时候听音乐, 那么用户启动浏览器和音乐播放器之间就有很强的关联性.

本文利用上述多种信息对用户行为进行建模, 分析用户使用应用程序的习惯, 预测用户未来使用每个 App 的可能性, 将最不可能使用的 App 终止, 清理出内存, 从而实现应用的高效管理. 实验结果表明, 本文提出的方法相比于 LRU 的方法可以有效降低应用程序重启率, 从而降低延迟和能耗、提升用户体验. 除此之外, 随着 MAX_HIDDEN_APPS 的逐渐变小, LRU 方法的命中率降低很明显, 而本文提出的方法降低幅度相对更缓慢, 在 MAX_HIDDEN_APPS 较小的时候可以达到 LRU 在 MAX_HIDDEN_APPS 较大的时候的效果, 使用本文提出的方法可以在内存中管理更少的 Apps, 从而节省内存管理开销.

后文分为以下几部分描述: 第二部分介绍相关工作, 第三部分介绍本文的系统和算法背景, 第四部分介绍本文提出的模型, 第五部分介绍实现在 Android 平台上的系统架构图, 第六部分实验分析, 第七部分总结本文的工作.

1 相关研究

1.1 手机用户行为分析

近年来, 很多研究者开始对手机用户的行为进行建模, 分析用户使用习惯来做一系列个性化优化. 一些学者[2-6]贡献了综合而全面的真实智能手机使用的分析报告, 涵盖了用户通用使用行为、电量交互、能耗、网络活动等方面. Baik[7]等人提出基于 App 的使用模式动态调整 cache limit, 从而在不降低缓存命中率的情况下让系统内存被更加高效的使用. 另一些学者[8-12]主要着眼于利用用户行为分析对电量、能耗的优化. 还有的学者[13-15]尝试分析用户使用习惯, 预测用户接下来可能启动的 App, 并将相关文件提前加载的内存中, 降低启动延迟, 提高用户体验. Zhang[16]等人分析用户使用习惯, 开发了个性化自适应的 App 桌面图标 UI, 预测用户接下来的 App 使用倾向, 据此管理 App 图标位置, 帮助用户快速找到所需 App.

本文与上述研究的不同之处在于, 上述研究中一部分研究着眼于分析, 而本文更强调系统优化. 另外, 与系统优化的部分研究相比, 它们主要着眼于电量、能耗管理以及应用程序预加载, 而本文着眼于智能手机系统的应用管理优化.

1.2 智能手机应用管理优化

关于智能手机应用管理, Android 系统采用基于 LRU 的 Task killing 机制, 但是 LRU 算法只考虑了最近使用 App 的情况, 并没有考虑用户的应用程序使用习惯, 并不能准确预测用户接下来可能启动的 App, 从而可能导致大量应用程序重启动, 带来延迟增加、能耗增加, 状态丢失等问题.

为了解决 LRU 的上述问题, Song[1]等人提出基于 Pattern 和 Cluster 的方法预测用户接下来启动各个 App 的概率从而进行应用管理. 然而他们只考虑了 app 间的使用关联模式并没有考虑时间、地点等场景信息. 而这些场景信息对于用户行为很有价值, 例如用户在办公室可能会使用一些办公类软件, 在家更多地使用游戏影音类软件, 而在户外可能使用地图类软件.

本文提出一种基于贝叶斯网络的智能手机应用管理优化方法, 综合利用用户 App 使用日志、情景信息、App 间关联等丰富的信息分析用户使用行为, 预测用户接下来启动各个 App 的概率, 从而进行应用管理.

2 论文背景

2.1 Android 应用管理

在 Android 的应用管理机制中, 一个应用程序往往由很多个 activities 组成. 当一个 activity 被启动时, 它会被压栈到活动栈的栈顶, 然后触发 on-create 阶段, 初始化接口、加载所需组件. 在 on-create 阶段之后依次是 on-start 阶段和 on-resume 阶段. 大多数情况下 on-create 阶段占用了启动过程中最长的时间. 这个 activity 会一直在前台运行直到启动了另一个 activity, 此时第一个 activity 会被切换到后台等待再一次启动. 如果一个后台 activity 再次被启动, 由于之前已经做过初始化工作了, 这次启动不会再执行 on-create 阶段.

包含 on-create、on-start、on-resume 三个阶段的完整的启动过程叫做“冷启动”[5], 不包含 on-create 阶段的启动过程叫作“热启动”. 前述的重启动是指冷启动. 后文交替使用“重启动”和“冷启动”两个术语.

活动栈的大小有限, 如果活动栈满的时候要启动新的 activity, 需要选择并删除活动栈中的一个 activity. Android 默认是采用 LRU 算法来选择要删除的 activity. LRU 算法假设用户最近使用过的 activity 有很高的概率会在不久的将来再次被启动. 基于 LRU 算法的删除流程如图 1 所示.

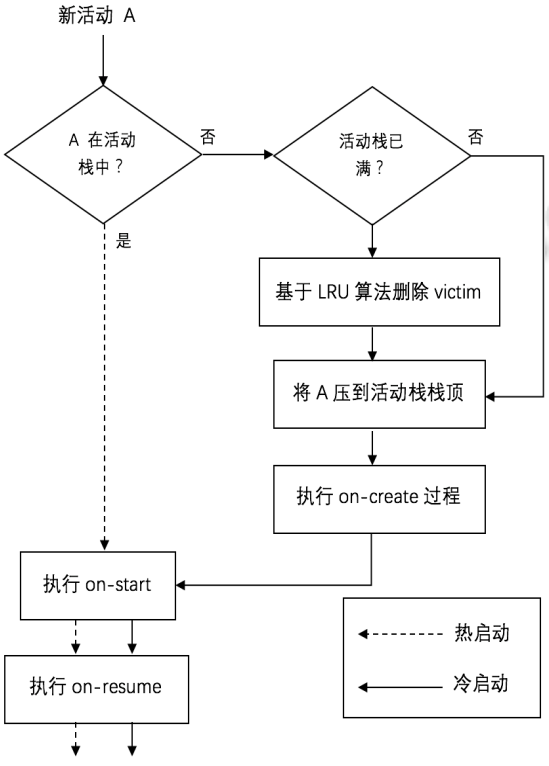

图 1 基于 LRU 的活动栈

然而基于 LRU 的方法只考虑了最近的应用程序使用情况, 并没有考虑用户的使用习惯以及各种情景信息, 不能准确预测用户未来的行为, 可能导致 kill 得不准从而造成较多的应用程序重新启动. 这些重新启动会带来延迟增加、能耗增加、状态丢失等问题, 降低了系统性能以及用户体验. 除此之外, LRU 的效果很依赖于 MAX_HIDDEN_APPS 的大小, MAX_HIDDEN_APPS 越大, 效果越好. 然而在内存中维护大量的后台进程本身也需要一些管理开销. 较大的 MAX_HIDDEN_APP 不仅会带来更多的内存管理开销, 甚至可能带来内存泄露的风险[1].

2.2 贝叶斯网络简介

贝叶斯网络[17](Bayesian network), 又称信念网络(Belief network), 或有向无环模型(Directed acyclic graphical model), 是一种概率图模型, 于 1985 年由 Judea Pearl 首先提出. 它是一种模拟人类推理过程中因果关系的不确定性处理模型, 其网络拓扑结构是一个有向无环图(DAG). 其中每个节点代表一个随机变量, 节点间的边代表变量之间的直接依赖关系.

若其中节点代表的随机变量集合为 $X = \{X_1, X_2, X_3, \ldots, X_n\}$, 随机变量 X_i 的“因”也就是父节点集合为 $Pa(X_i)$, 那么联合概率分配为:

$$P(X_1,X_2,\ldots,X_n) = P(X_1)*P(X_2)*\ldots\ldots*P(X_n \mid X_1,X_2,\ldots,X_{n-1}) = \prod_{i=1}^{n} P(X_i \mid Pa(X_i))$$

在给定”因变量”的条件下, 该节点会与其非因变量条件独立. 如果联合概率分配的相互依赖的数目很少时, 使用贝叶斯网络可以节省相当客观的存储容量. 例如, 若想将 10 个二值变量存储为条件概率表达式, 总共需要计算 $2^{10} = 1024$ 个值. 但是如果这 10 个变量中每个变量的“因变量”都不超过三个, 那么贝叶斯网络的条件概率表最多只需要计算 $10 * 2^3 = 80$ 个值即可. 另外, 利用贝叶斯网络, 可以轻易得知各随机变量间是否条件独立.

3 基于贝叶斯网络的应用管理方法

3.1 情景信息

通过第 2.1 节的描述可知, 为了降低延迟和能耗、提升用户体验, 降低冷启动次数也就是应用程序重启动次数变得尤为重要. 而降低冷启动的关键在于一个好的 victim 选择方法, 将准确预测用户在不久的将来

可能会启动的应用程序.

基于 LRU 的方法仅仅考虑了用户最近使用 App 的日志这一单一信息, 并未考虑情景信息以及 App 间的关联, 因此可能并不能准确预测用户将来的使用倾向, 从而带来大量的应用程序重启动.

本文综合考虑了用户 APP 使用日志、App 间关联、情景信息三方面的信息, 可以得到比 LRU 更加准确的预测结果, 从而降低冷启动次数, 达到降低延迟和能耗, 提高系统性能和用户体验的目的.

情景信息主要是指用户所处的环境状态, 例如时间、地点等等. 情景信息对于预测用户接下来的 App 使用行为具有重要影响. 本文的实验基于 LiveLab[18] 数据集, 通过对数据集的分析, 提取情景信息如下.

表 1 情景信息

Context	Value
Time of day	Morning, Afternoon, Evening
Day of week	Weekdays, Weekends
Location	Office, Home, Other
LastApp	Last used App

3.2 地点识别算法

用户使用手机过程中 GPS 并不总是开启的, 因此当 GPS 关闭时需要利用其他信息识别用户所在地点. 针对这个问题, Zhang[16]等人提出一种基于决策树的地点识别算法. 本文模型的地点识别模块使用的就是该文所提出的方法. 该算法主要利用了两部分信息, 一是 wifi 信息, 一是 cell 信息. 算法假设大部分用户常去室内地点有 wifi 连接, 因此 wifi 信息可以用来标识用户常用的室内地点, 而 cell 信息则用来标识用户常用的室外地点.

3.3 基于贝叶斯网络的 App 启动概率预测模型 BNLP

为了研究各种情景信息(“因”)对用户下一个可能启动的 App 的影响(“果”), 本文构建了一个贝叶斯网络模型 BNLP(Bayesian network-based launch predictor)来预测用户启动某个 App 的概率. 贝叶斯网络的结构如图 2 所示.

该贝叶斯网络假设某 App 即将被启动的概率依赖于 time of day, day of week, location 以及 lastApp. 而 location 则依赖于 time of day 和 day of week. 每当后台进程数达到 MAX_HIDDEN_APPS 并且需要启动新的 App 时, 模型会根据下面的公式对所有后台 App 计算一个分值, 该分值度量了该 App 即将被打开的概率大小. 模型选择得分最低的 App 终止, 清理出内存.

$$Score(A) = P(A|D,T,L,A') = P(A|A')*P(A|D,T,L)$$

整个 BNLP 模型从原始数据输入到 Score 输出的流程如图 3 所示.

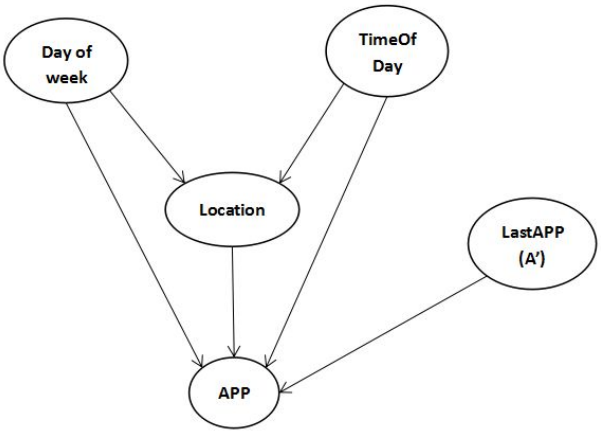

图 2 贝叶斯网络结构

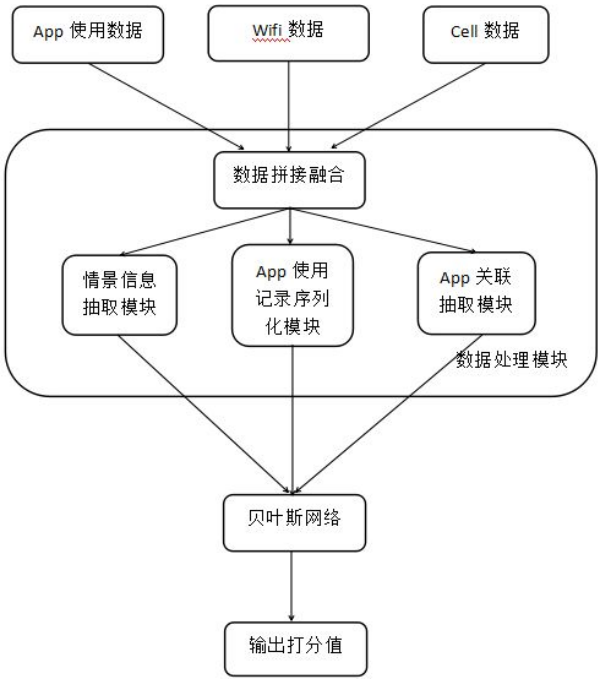

图 3 BNLP 模型预测流程

Appusage 数据、wifi 数据、cell 数据首先进入到数据拼接模块, 进行数据拼接与融合. 然后输出给情景信息抽取模块进行情景信息的抽取, App 使用记录序列化模块进行使用日志序列化, App 关联抽取模块抽取 App 间共现关联. 数据拼接融合模块加上后面三个模块构成了整个数据处理模块. 最后三方面的数据作为贝叶斯网络的输入, 贝叶斯网络整合这些信息最

终输出相应的预测值.

3.4 基于 BNLP 的 Task killer

当后台进程数达到 MAX_HIDDEN_APPS 并且需要启动新进程的时候会触发基于 BNLP 模型的 Task killing 机制. 基于 BNLP 模型选择和终止应用程序的流程如图 4 所示.

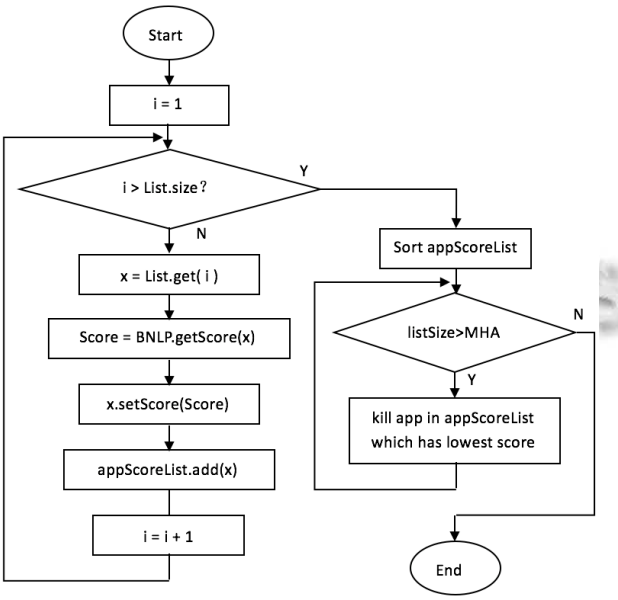

图 4　基于 BNLP 选择和终止应用程序

由图 4 可知, 一旦触发基于 BNLP 的 Task killing 机制, 模型会对所有后台应用程序 List 中的应用程序计算一个分值 score, 该分值度量该应用程序被启动的概率. 将所有应用程序根据得分值排序, 只要后台应用程序数大于 MHA, 即 MAX_HIDDEN_APPS, 则从排好序的应用程序中选择得分最低的应用程序终止, 清理出内存.

3.5 历史窗口大小 windowSize

贝叶斯模型中所用到的条件概率都是基于一定长度的历史数据计算得到的, 而这个长度大小是 windowSize(天), windowSize 设置得大一些能够得到更加充分的信息, 模型能够更好的建模用户随时间变化不明显的长期的使用习惯. 而 windowSize 设置得较小可以敏锐捕捉到用户随时间短期内变化明显的使用习惯. 例如, 用户在以前喜欢玩一款游戏类 App, 但一周后喜欢上了另一款游戏类 App. 较小的 windowSize 能够使得模型对用户短期兴趣变化有更好的适应能力.

4　系统架构

最终我们将 BNLP 模型实现在了 Android 手机上, 系统架构图 5 所示. 整个系统主要包括信息记录模块、BNLP 预测模块、监控与应用管理模块三大模块.

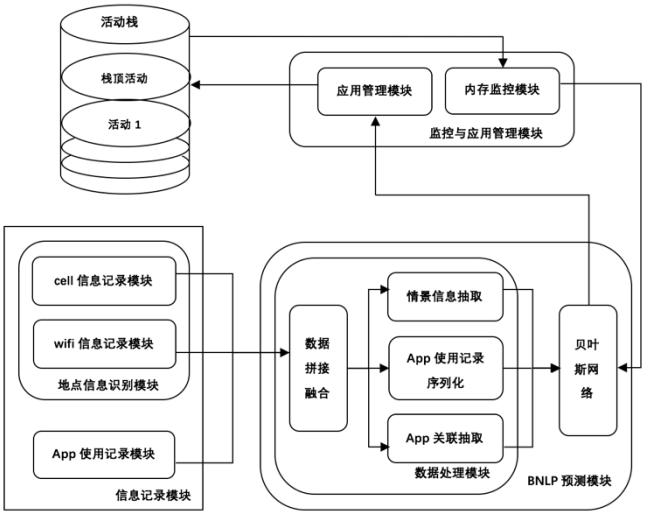

图 5　系统架构

信息记录模块负责定时记录 cell 信息、wifi 信息以及 App 使用信息. 其中的地点信息识别模块利用 cell 信息和 wifi 信息记录模块得到的 cell 信息和 wifi 信息识别出地点信息.

BNLP 预测模块将信息记录模块传入的数据作为输入, 进入到数据拼接融合模块, 进行三类信息的拼接整合, 然后整合后的数据会分别进入到后续三个子模块进行情景信息抽取、App 使用记录序列化以及 App 的前后共现关联抽取. 这四个子模块构成了 BNLP 预测模块中的数据处理模块. 数据处理完之后情景信息、App 使用记录以及 App 间的关联会输出给贝叶斯网络, 贝叶斯网络会给所有的后台 App 打分(后台 App 链表由内存监控模块传入), 将得分最低的 App 信息返回给监控与应用管理模块.

监控与应用管理模块包括内存监控模块和应用管理模块. 内存监控模块会实时监控内存的使用状态, 当需要终止一个 App 的时候, 调用 BNLP 预测模块, 并给贝叶斯网络传入当前内存中的后台 App 链表. 当 BNLP 预测模块得到预测结果后, 将需要终止的 App 信息返回给应用管理模块, 由应用管理模块去终止相应 App.

5 实验

5.1 实验说明

本文将基于贝叶斯网络的应用管理方法实现在了 Android 手机上, 具体的系统架构参见第 4 部分内容. 为了在其他条件相同的情况下公平地比较各个模型的效果, 本文实验设置为利用 LiveLab[19]数据集在我们的系统上做了重播(Replay). 按照该数据集的 App 使用序列启动对应的 Android 程序以模拟用户在 Android 系统上真实地启动 App, 接着采用不同的方法进行应用管理, 对比分析各个方法的实验效果. 另外, 为了快速得到结果, 我们将 App 使用序列中的每次 App 使用时间限制在 10 秒内.

5.2 实验数据集

本文使用的 LiveLab 数据集, 是美国 Rice University 收集并公开的 34 个学生一年时间内的真实手机使用记录. 除此之外该数据集还包括非常丰富的手机性能状态数据, 具体参见 LiveLab 官网. 数据集统计数据如表 2 所示.

表 2 数据集统计信息

数据类型	数据条数
App 使用日志	1, 099, 595
连接 wifi 数据	38, 439, 198
可用 wifi 数据	654, 565
Cell 数据	321, 827

由于在 LiveLab 数据集的 app 使用日志中并没有 wifi 信息和 cell 信息, 需要利用用户和时间信息与 wifi 信息数据以及 cell 信息数据建立联系. 并且三个表格时间粒度不同, 因此我们进行了信息融合, 对于没有的 wifi 信息或者 cell 信息用时间最近的代替.

5.3 评价指标&实验设置

根据数据集整理出用户在不同时间使用不同 Apps 的真实使用序列, 利用 LRU 算法对后台的 Apps 进行管理, 记录管理后每个时间点后台存在的 App, 如果真实使用的 App 没在内存中, 算一次重启. 统计完整个使用序列就得到了 LRU 算法的重启次数, 除以总记录数就得到了 LRU 算法的重启率. 同样地, 使用 BNLP 模型对后台的 Apps 进行管理, 可以得到 BNLP 模型的重启率, 比较两个算法的重启率即可以比较二者的优劣. 重启率定义如下:

重启率 = 重新启动的次数 / 总次数

具体而言, 一个计算重启率的例子如图 6 所示.

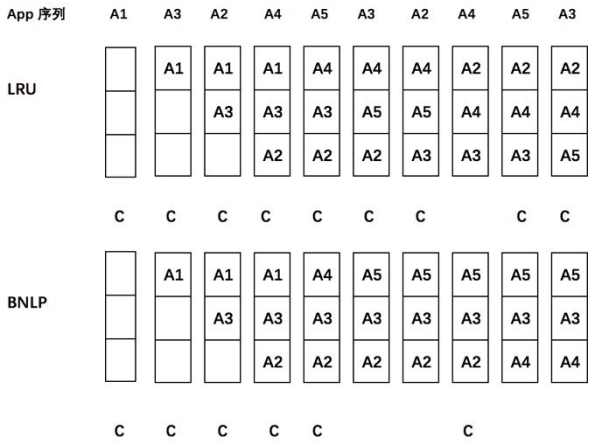

图 6 应用管理过程

假设 MAX_HIDDEN_APPS = 3, 第一排是 App 的使用序列, 后面是分别基于 LRU 算法和 BNLP 算法时内存中存放的后台进程, “C”(cold start)代表该时刻启动的 App 没有在内存中, 因此需要一次冷启动(重启动).

由图 6 的例子可以看出基于 LRU 的 Task killer 带来了 9 次重启动, 重启率为 9/10 = 90%. 而基于 BNLP 的 Task killer 带来了只 6 次重启动, 重启率为 6/10=60%.

这主要是因为 LRU 只考虑最近的 App 使用情况, 终止的应用程序可能很快又被用户启动. 而 BNLP 能够发现 App 间的使用关联模式, 例如图中的“A3->A2”以及“A2->A4”这样的模式, 并据此进行合理的预测与管理, 从而降低了重启动次数.

5.4 用户差异实验

5.4.1 用户使用 App 的差异

用户间使用 App 的数量、常用 App 数都有很大差异, 而由于这些差异导致不同模型效果对于不同用户也有一定差异.

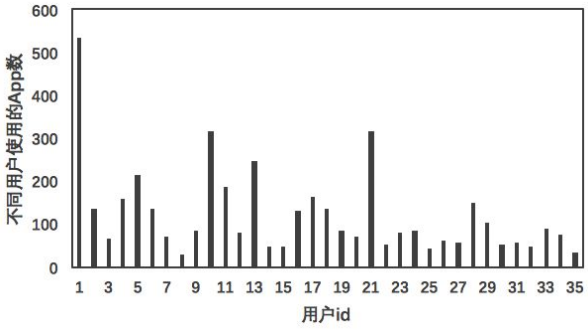

图 7 不同用户使用的总 App 数

图 7 展示了不同用户使用的总 App 数, 由该图可以看出不同用户使用的 App 数差异非常大, 使用 App 少的用户只有 3、40 个 Apps, 而使用 App 多的用户可以达到 500 多个 Apps.

图 8 展示了不同用户 top K 个 Apps 的使用次数占总次数的比值. 可以看出不同用户差异明显. 但是总体上看用户常用的 App 数并不多, 基本上 top 6 到 12 个就已经占了总使用次数的 90% 以上. 当 MAX_HIDDEN_APPS 的值较大时, 用户常用 Apps 基本上都能在内存中. 这也是为什么 LRU 算法在 MAX_HIDDEN_APPS 较大的时候可以取得不错结果.

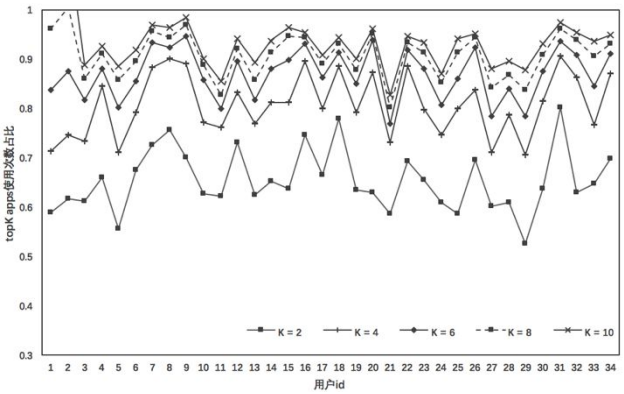

图 8　不同用户 top K 个 apps 使用次数占比

另外, top K 个 apps 的使用次数占比和用户使用的总 App 数基本上是一个负相关的关系. 也就是说使用 app 多的用户 topK 个 apps 使用次数占比相对较低, 如用户 1、5、10、13、17、21. 而 3、8、14、32 等用户使用的总 app 数比较少, 从而 top K 个 apps 的占比相对就更大了. 而一些特殊的用户例如 25, 使用的总 app 少, top K apps 的占比也少, 这类用户使用各个 app 的频率相对比较均匀.

5.4.2 模型对不同用户的预测结果

由 5.4.1 小节的分析可知用户使用 App 的数量、常用 App 数上有很大差异, 那么可能在一定程度上模型对不同用户的预测表现也会有一些差异.

由图 9 可以看出对所有用户而言, BNLP 算法效果均优于 LRU 算法. 另外, 不同用户重启率差别很大, 有的用户即使使用 LRU 算法也具有较低的重启率, 可能是因为这类用户常用 App 数较少, App 使用习惯跟时间地点等情景信息关系不大. 除此之外, 对不同用户 BNLP 模型的提升效果也不尽相同. 例如对 2、3、5、6、11 等用户而言, 提升效果明显, 而对 4、7、9、12、20 等用户而言, BNLP 和 LRU 算法效果相差不多.

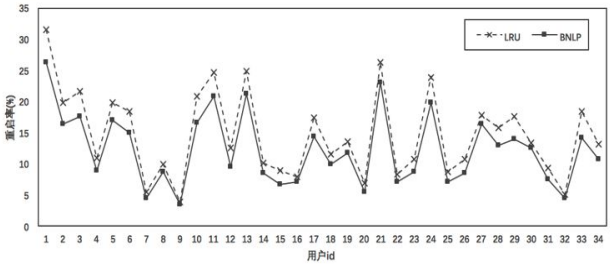

图 9　模型对不同用户行为预测结果图 (MAX_HIDDEN_APPS=7)

综上所述, 不同用户使用 App 的行为和习惯都很不相同, 无法用一个统一的不考虑个性化的方法去很好的对应用进行管理. 相反的, 对用户行为进行建模的方法则能够描绘出用户不同的使用习惯, 从而更加有效地管理应用.

5.5 重启率比较

前述模型中有两个条件概率, P(A|A')和 P(A|D, T, L). 我们可以将它们看做是对 App 启动的两种不同的影响因素, 前者代表 App 间的关联关系对 App 启动的影响, 后者代表时间和地点情景对 App 启动的影响. 图 10 展示了几种不同影响因素作用下重启率随 MAX_HIDDEN_APPS 变化曲线, 其中重启率是 34 位用户的平均重启率.

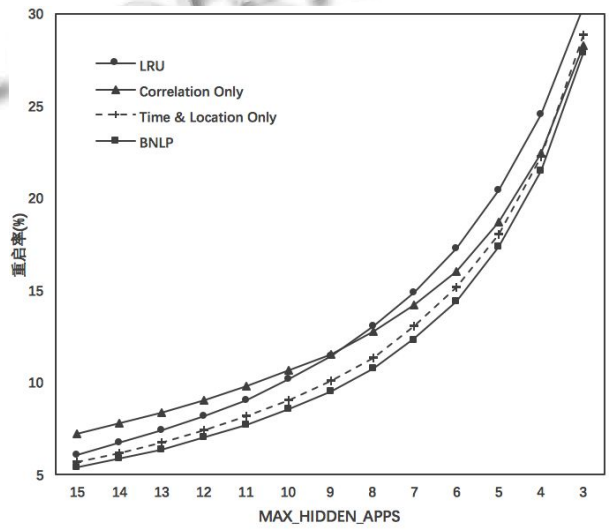

图 10　重启率随 MAX_HIDDEN_APPS 变化

由图 10 可以看出:

① 随着 MAX_HIDDEN_APPS 变化，贝叶斯网络模型 BNLP 的重启率始终低于 LRU 算法. 这是因为 BNLP 考虑了 App 使用日志、情景信息、App 间关联三方面的丰富的信息，可以更加准确地预测用户接下来的使用倾向，因而具有比 LRU 更好的效果.

② 只考虑 App 间关联作用下的重启率先是高于 LRU，然后逐渐趋向 LRU 并最终低于 LRU. 这是因为用户常用 App 数本身不多，当 MAX_HIDDEN_APPS 设置得较大时，LRU 的方法就能具有还不错的效果，而 App 间关联在内存中 App 数较多的时候可能会存在一些噪声.

③ 两个影响因素独立作用下，总体上来看重启率都是低于 LRU 的，这说明情景信息对于预测用户使用倾向确实是有帮助的. 只考虑时间和地点影响作用下的重启率始终低于只考虑 App 间关联作用下的重启率. 这说明对 LiveLab 数据集的这 34 个用户而言，平均情况下时间和地点对预测用户行为更有意义.

④ 整合了两个影响因素的贝叶斯网络重启率低于影响因素单独作用的重启率，又进一步低于 LRU 的重启率. 说明整合多方面信息的贝叶斯网络确实能更加有效地预测用户的使用倾向.

⑤ 另外, BNLP 模型可以在 MAX_HIDDEN_APPS 较小的时候达到和 LRU 在 MAX_HIDDEN_APPS 较大时的重启率，因此可以将 MAX_HIDDEN_APPS 设置得小一些，减少内存 App 管理开销.

根据 5.4 的分析可知，对大多数用户而言，将 MAX_HIDDEN_APPS 设置为 7 是一个合理的选择. 当 MAX_HIDDEN_APPS = 7 的时候，几种方法的重启率如表 3 所示.

表 3 重启率比较(MAX_HIDDEN_APPS = 7)

方法	重启率
LRU	14.89%
Correction only	14.16%
Time & Location only	13.01%
BNLP	12.33%

由该表可以看出后三种方法均取得了比 LRU 更低的应用程序重启率，当 MAX_HIDDEN_APPS = 7 的时候, BNLP 相对比 LRU 算法重启率降低了 17.2%.

5.6 windowSize 对预测效果的影响

如 3.5 小节所述，较大的 windowSize 可以给模型提供更加充分的数据，以对用户长期的随时间变化不大的兴趣进行建模，而较小的 windowSize 则能更敏锐地捕捉用户短时间内的兴趣变化. 图 11 展示了模型效果随着 windowSize 的变化结果.

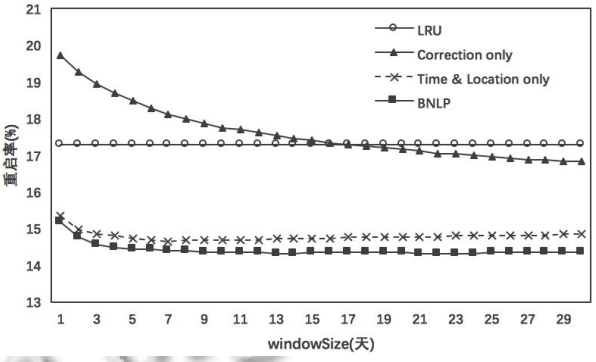

图 11 模型效果随 windowSize 变化曲线 (MAX_HIDDEN_APPS=6)

由图 11 可以看出 LRU 的重启率不随 windowSize 变化，这是因为 LRU 只考虑最近使用情况，并不需要历史数据. 因而也无法捕捉用户长期或者短期的兴趣. 只考虑 App 间关联重启率随 windowSize 增加而降低，说明历史数据越多，利用 App 间关联信息进行预测越准确. 只考虑时间和地点时，重启率随 windowSize 增加先降后趋于稳定并有轻微回升，这说明利用时间和地点进行预测需要一定量的历史数据，此外过多的历史数据也可能带来一些噪声. 最后在两个影响因素的共同作用下，BNLP 模型的重启率先降低后逐渐趋于稳定.

6 结语

从前文的描述可以看出，本文提出的 BNLP 模型整合了 App 使用日志、情景信息、App 间关联三方面的丰富的信息，并且利用贝叶斯网络对于各因素对 App 启动间的因果关系进行了合理的假设和建模，能够比 LRU 算法更加准确地预测用户接下来的 App 启动倾向. 最终在 LiveLab 数据集上的实验表明，基于 BNLP 算法的 Task killer 能够比基于 LRU 的 Task killer 带来更低的重启率，从而达到降低延迟和能耗，提高用户体验的目的.

参考文献

1 Song W, Kim Y, Kim H, et al. Personalized optimization for Android smartphones. ACM Trans. on Embedded Computing

Systems (TECS), 2014, 13(2s): 60.

2 Shye A, Scholbrock B, Memik G, et al. Characterizing and modelinguser activity on smartphones: Summary. ACM SIGMETRICS Performance Evaluation Review. ACM, 2010, 38(1): 375–376.

3 Falaki H, Mahajan R, Kandula S, et al. Diversity in smartphone usage. Proc. of the 8th International Conference on Mobile Systems, Applications, and Services. ACM. 2010. 179–194.

4 Do TMT, Blom J, Gatica-Perez D. Smartphone usage in the wild: A large-scale analysis of applications and context. Proc. of the 13th International Conference on Multimodal Interfaces. ACM. 2011. 353–360.

5 Xu Q, Erman J, Gerber A, et al. Identifying diverse usage behaviors of smartphone apps. Proc. of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference. ACM. 2011. 329–344.

6 Kang JM, Seo S, Hong JWK. Usage pattern analysis of smartphones. Network Operations and Management Symposium (APNOMS), 2011 13th Asia-Pacific. IEEE. 2011. 1–8.

7 Baik K, Huh J. Balanced memory management for smartphones based on adaptive background app management. The 18th IEEE International Symposium on Consumer Electronics (ISCE 2014). IEEE. 2014. 1–2.

8 Shye A, Scholbrock B, Memik G. Into the wild: Studying real user activity patterns to guide power optimizations for mobile architectures. Proc. of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture. ACM. 2009. 168–178.

9 Kravets R, Krishnan P. Application-driven power management for mobile communication. Wireless Networks, 2000, 6(4): 263–277.

10 Ra MR, Paek J, Sharma AB, et al. Energy-delay tradeoffs in smartphone applications. Proc. of the 8th International Conference on Mobile Systems, Applications, and Cervices. ACM. 2010. 255–270.

11 Könönen V, Pääkkönen P. Optimizing power consumption of always-on applications based on timer alignment. 2011 Third International Conference on Communication Systems and Networks (COMSNETS). IEEE. 2011. 1–8.

12 Zhang L, Tiwana B, Qian Z, et al. Accurate online power estimation and automatic battery behavior based power model generation for smartphones. Proc. of the eighth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis. ACM. 2010. 105–114.

13 Huang K, Zhang C, Ma X, et al. Predicting mobile application usage using contextual information. Proc. of the 2012 ACM Conference on Ubiquitous Computing. ACM. 2012. 1059–1065.

14 Esfahbod B. Preload--An adaptive prefetching daemon[Thesis]. University of Toronto, 2006.

15 Yan T, Chu D, Ganesan D, et al. Fast app launching for mobile devices using predictive user context. Proc. of the 10th International Conference on Mobile Systems, Applications, and Services. ACM. 2012. 113–126.

16 Zhang C, Ding X, Chen G, et al. Nihao: A predictive smartphone application launcher. Mobile Computing, Applications, and Services. Springer Berlin Heidelberg, 2013: 294–313.

17 Pearl J. Bayesian networks: A model of self-activated memory for evidential reasoning. University of California (Los Angeles). Computer Science Department, 1985.

18 LiveLab traces. http://livelab.recg.rice.edu/traces.html.