中国科学: 信息科学 2016 年 第 46 卷 第 6 期: 677–697
SCIENTIA SINICA Informationis

《中国科学》杂志社
SCIENCE CHINA PRESS

论 文

移动计算环境下基于动态上下文的个性化 Mashup 服务推荐

何伟[①②*], 崔立真[①②], 任国珍[①], 李庆忠[①②], 李婷[①]

① 山东大学计算机科学与技术学院, 济南 250101
② 山东大学山东省软件工程重点实验室, 济南 250101
* 通信作者. E-mail: hewei@sdu.edu.cn

收稿日期: 2015–06–17; 接受日期: 2015–08–20; 网络出版日期: 2016–05–26
国家自然科学基金 (批准号: 61572295, 61303085)、科技部创新方法专项 (批准号: SQ2015IMC600006) 和山东省自然科学基金 (批准号: ZR2014FM031, ZR2013FQ014) 资助项目

摘要 移动互联网的快速发展和智能终端数量的迅速增长使得 Internet 服务更加密切地融入到人们的生活中, 面向非专业用户的 Mashup 服务在移动计算环境下有着更广泛、迫切的需求, 同时, 用户行为过程中的上下文动态不确定性、用户行为模式复杂性和个性化特征等因素对传统 Mashup 技术提出了挑战. 针对目前 Mashup 技术和研究中存在的问题, 本文将关注点从离散的服务推荐扩展到 Mashup 构造与运行过程, 提出一种新的基于迭代式自主构造的 Mashup 服务组合和运行模式, 主要思想是对大量离散、孤立的历史执行轨迹进行挖掘处理, 建立上下文相关和偏好相关的用户行为选择概率模型, 为构造优化的个性化 Mashup 组合方案提供支持. 分析和模拟实验表明, 面对移动环境上下文动态性和普适性特点, 本文方法能够有效简化 Mashup 构造过程中用户参与的复杂性, 减少对用户专业知识的依赖; 同时, 在用户行为模式预测和 Mashup 服务构造与推荐方面能够获得比其他服务推荐或组合方法更好的效果.

关键词 移动计算环境 上下文 Mashup 服务推荐 协同过滤

1 引言

随着 Internet 上基于 Web 协议的 API 数量不断增长, 面向普通用户提供快速复合服务的 Mashup 技术发展迅速, 众多知名企业推出了 Mashup 商业或实验性平台, 如 Yahoo Pipes, Google Mapplets, Microsoft PopFly 等, 为用户提供可视、简单易用的交互界面; 另一方面, 当前 Mashup 应用和技术面临着一些公认的问题, 例如, 如何帮助用户从大量可用服务中发现合适的组件; 要求普通用户必须具备一定的专业知识, 包括理解 Mashup 组件的输入输出、能够规划并构造满足需求的复合服务等 [1]. 针对类似问题, 研究者们开展了大量有效的工作 [2~8].

近年来, 移动互联网的快速发展和智能终端数量的迅速增长, 使得 Internet 服务更加密切地融入到人们的生活中, 为用户带来更好的体验, 面向非专业用户的 Mashup 服务在移动互联网等普适计算

引用格式: 何伟, 崔立真, 任国珍, 等. 移动计算环境下基于动态上下文的个性化 Mashup 服务推荐. 中国科学: 信息科学, 2016, 46: 677–697, doi: 10.1360/N112014-00252

© 2016《中国科学》杂志社

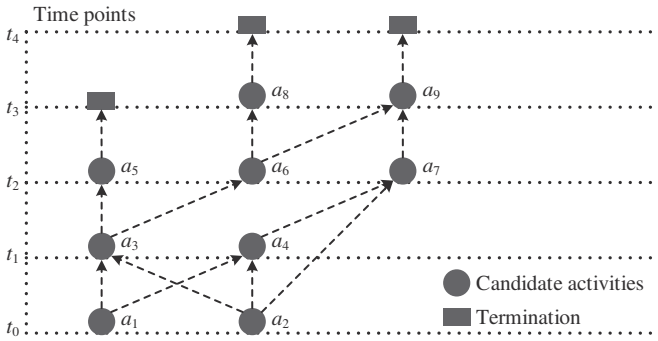

图 1　可能的 Mashup 多组合方案

Figure 1　Multiple potential solutions for a mashup

环境下有着更广泛、迫切的需求, 然而, 移动计算环境的时空动态性、普适性等特征也增加了 Mashup 服务构建的难度和复杂性:

• 用户行为的过程特征复杂性. 移动计算环境下用户行为的过程特征更加明显, 呈现出控制流驱动的过程化特点. 传统 Mashup 通常是数据流驱动的组合服务, 预先构建模式并绑定 Web 组件或数据源, 最后执行完成; 而在移动互联网环境下, 一个 Mashup 的构建和执行过程往往伴随着用户的自然行为, 具有过程性[1)]和交互性特点, 构成 Mashup 的 Web 组件之间更接近于控制流驱动的关系, 对于非专业用户来说构造这样一个 Mashup 更加困难.

• 上下文感知复杂性. 在移动互联网环境下, 用户所处上下文环境随时发生变化, 时空复杂性和动态性十分明显, 即使对同一用户, 不同时刻或不同地点的 Mashup 方案在一个新的上下文环境中也可能不再适合, 传统 Mashup 预先规划组合结构的方式已无法适应这种变化, 需要通过感知不同场景, 即时规划用户行为并推荐最为合适的 Web 组件或服务.

• 用户即时偏好的复杂性. 用户习惯、偏好等因素决定着用户行为的选择, 即使在同一场景中, 不同用户的行为选择偏好也不同, 而移动性增加了用户偏好感知的复杂性. 移动互联网环境下的 Mashup 在行为规划、服务推荐中需要考虑个性化即时偏好的影响, 以更好地改善用户体验.

例如, 对于一个面向就餐的 Mashup 服务过程, 用户活动可能包括查找餐馆、预订、到达餐馆、点餐、支付、评价等, 涉及到的可用互联网服务包括 LBS 位置服务、地图服务、导航服务、预定服务、点餐服务、支付服务、评价服务等, 这些服务的规划和选择应与用户在这一过程中行为模式尽可能一致. 然而, Mashup 服务的过程化特征使得其中的用户活动不再孤立, 系统对用户行为模式的感知和预测变得困难, 更为重要的是, 在移动计算环境中, 用户所处上下文具有动态性和不可预见性, 同时考虑到用户偏好的影响, 进一步增加了 Mashup 服务规划和基于用户行为推荐服务的困难. 比如, 在基于宽带网络的桌面环境下, 用户可能倾向于包含从搜索餐馆到点餐和支付的规划和执行; 在基于智能终端的移动环境下, 只包含位置、地图或导航服务的规划可能更为合适; 而用户上下文环境转换的不确定性, 使得 Mashup 构建和执行过程存在即时性和动态性特征, 从而更加复杂. 因此, 如何基于动态上下文帮助用户规划出合适的个性化 Mashup 模式, 减少构造过程对用户专业知识的依赖, 比仅仅推荐合适的 Web 组件更加重要, 也是对传统 Mashup 技术的挑战.

事实上, 针对特定的用户需求, 必然存在多个可行的 Mashup 服务规划方案, 图 1 列出了满足某

1) 称之为 Mashup 而非流程, 是因为业务流程以预定义的结构化或半结构化模型为基础, 具有相对严格的模式定义、实例执行、控制和审计过程, 而 Mashup 是基于互联网组件、直接面向最终用户提供 ad-hoc 复合服务的轻量级方法.

特定需求的可能的组合方案. 例如, 活动序列 $\langle a_1, a_3, a_5\rangle$ 和 $\langle a_2, a_4, a_7, a_9\rangle$ 都能够满足需求, 但是在不同上下文环境下带给用户的体验也不相同. 传统环境下的 Mashup 模式需要在最初时刻 t_0 规划好, 但是, 此时并无法确定后续时刻 (t_1, t_2, t_3) 用户所处上下文环境, 将导致预先构造的 Mashup 模式难以获得良好的效果.

目前, 针对 Mashup 模式及技术的研究侧重于对构成 Mashup 组件的搜索和推荐 [1], 研究者提出了很多有效的方法, 包括基于组件 QoS 推荐 [3]、基于语义推荐 [4,5]、基于协同过滤推荐 [6]、基于社交网络推荐 [7] 等, 这些组件推荐方法的前提是用户已经预先规划了 Mashup 模式. 但是, 在移动计算环境下, 预先规划一个明确的 Mashup 结构并不可行, 既无法适应用户上下文的动态性和不确定性, 对非专业用户来说也过于困难; 在上下文感知的服务组合领域, 研究者提出了基于服务上下文匹配的组合框架 [9], 基于规则的上下文感知服务组合方法 [10], 上下文感知的普适服务组合 [11,12], 基于规划的组合方法 [13,14] 等, 主要研究内容集中于服务组合中的上下文感知方法与模型, 个性化服务推荐方法, 以及组合模型的生成算法等, 而很少关注服务执行过程中用户行为模式的挖掘与预测, 特别是上下文环境变化、用户偏好等因素对用户所期望活动序列的影响, 而后者对于 Mashup 服务过程中用户体验的改进更为重要, 同时也比单纯的服务推荐更加困难.

针对移动计算环境下 Mashup 应用和技术面临的问题, 本文将关注的焦点从 Mashup 组件推荐扩展到伴随用户行为过程的 Mashup 构造与运行周期, 提出并研究一种面向动态上下文和用户偏好的 Mashup 自主构造和运行模式, 主要思想是对大量离散、孤立的历史执行轨迹进行挖掘处理, 从中发现潜在的基于上下文和用户特征的行为模式, 建立上下文和偏好相关的用户活动选择概率模型, 为构造并推荐优化的个性化 Mashup 组合方案提供支持. 本文提出的 Mashup 自主构造和运行模式不再依赖用户利用自身专业知识预先规划 Mashup 组合结构, 而代之以一种渐进的 "构造 – 执行" 迭代式过程, 伴随执行过程中感知的即时场景、活动轨迹和用户偏好, 逐步规划出下一步行为并推荐 Web 组件, 最终实现整体目标. 本文主要贡献如下: 1. 提出基于日志挖掘的 Mashup 模式重构方法, 从大量离散的执行实例中提取出上下文和偏好相关的用户行为模式, 通过用户活动选择概率模型映射出潜在的 Mashup 组合模式; 2. 对于历史日志中用户活动所处上下文不完整或缺失的情况, 提出一种基于 Bayes 网络的上下文推理方法, 从而改进活动推荐的准确度; 3. 基于上述方法, 提出基于上下文的个性化 Mashup 自主构造方法, 在该方法中, Mashup 构造和执行成为一个迭代的过程, 每次迭代被分为两个阶段: 模式构造阶段和组件推荐与绑定阶段.

本文其余部分组织如下: 第 2 节给出问题场景描述和相关概念, 第 3 节描述系统整体模型, 第 4 节给出基于日志挖掘的 Mashup 模式抽取方法, 第 5 节详细讨论基于条件概率模型的两阶段 Mashup 构造方法, 第 6 节介绍了模拟实验情况, 第 7 节介绍了相关工作, 第 8 节对本文进行总结, 提出下一步需要完善的工作.

2 问题模型

本文所讨论的移动互联网环境下的 Mashup 应用场景如图 2 所示, 包括组件域、Mashup 域、用户及上下文, 组件域包含了分布于 Internet 的、可被识别和绑定的 Web-API 组件, Mashup 域包括 Mashup 构造与执行引擎、组件注册库和历史执行日志等相关数据; 用户提出 Mashup 需求并在执行过程进行必要的交互, 而上下文是指影响用户行为选择的环境信息, 包括用户属性、设备属性、软件环境、网络状况等.

针对此应用场景, 本文关注 Mashup 构造与执行过程中的核心问题, 即上下文和用户偏好相关的

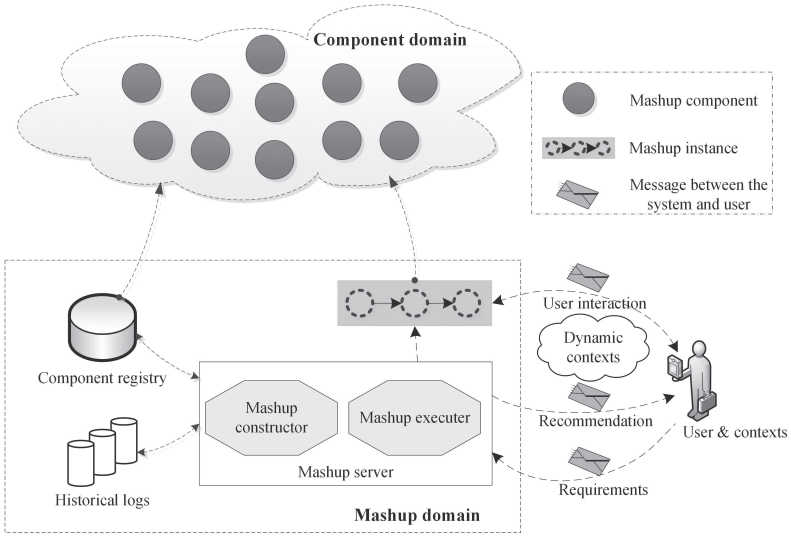

图 2　基于上下文和用户偏好的 Mashup 场景

Figure 2　The context- and user preference-based mashup scenario

Mashup 组合结构自主构造与服务推荐, 下面给出其中相关的术语定义.

定义1 (Mashup 组件)　构成 Mashup 的原子服务, 是 Web-API 服务实体在操作级别的抽象, 表示为元组: mc = ⟨Op, In, Out, Desc⟩, 其中 Op 表示操作名称, In = {in_1, in_2, ...} 和 Out = {out_1, out_2, ...} 分别是输入和输出参数集合, 每个参数表示为一个元组: p = ⟨name, type, req⟩, $p \in \text{In} \cup \text{Out}$, 其中 req∈{OPTIONAL, REQUIRED} 表示参数是否必选. Desc 表示当前组件的标注集合: Desc={anno_1, anno_2, ...}.

在上下文感知 (context-aware) 技术中, 上下文通常是指时间、位置、温度等用户相关的物理状态 [15], 本文中上下文的概念范畴更大一些, 还包括 Mashup 执行过程中的抽象信息, 比如用户已完成的活动轨迹等.

定义2 (Mashup 上下文)　上下文参数的集合: mctx={mcp_1, mcp_2, ...}, 其中每个参数表示为一个元组: mcp_i=⟨pn, pt, pv⟩, pn, pt, pv 分别表示名称、数据类型和参数值.

Mashup 实例表示一次完整的 Mashup 执行轨迹, 包括涉及到的所有组件、组件绑定与执行上下文、组件执行序列及相关用户信息. 离散的 Mashup 实例集合构成了 Mashup 日志.

定义3 (Mashup 实例)　一个 Mashup 实例表示为三元组: mi = ⟨usr, req, ts⟩, 其中 usr 表示用户, req 表示用户需求 req = ⟨In, Out, Desc⟩, In = {in_1, in_2, ...}, Out = {out_1, out_2, ...}, Desc = {kw_1, kw_2, ...} 分别是输入、输出和描述关键字集合. ts 表示任务序列: ts = ⟨t_1, t_2, ..., t_n⟩, 其中每个任务 t_i ($1 \leqslant i \leqslant n$) 可表示为一个元组 t_i=⟨mc_i, ctx_i⟩, ctx_i, mc_i 分别表示任务所在上下文和绑定的 Mashup 组件. 在该任务序列中, 组件 mc_i 所需输入参数来自 mc_{i-1}, mc_{i-2}, ..., mc_1 中 1 个或多个组件的输出 ($1 \leqslant i \leqslant n$).

对于任意 Mashup 实例 mi, 如果满足

$$\bigcup_{j=1}^{|\text{mi.ts}|} \text{mi.ts}[j].\text{mc}_j.\text{Out} \supseteq \text{mi.req.Out}, \tag{1}$$

则称 Mashup 是可行的, 这一约束不同于传统意义的组合服务, 用户需求可由构成 Mashup 的多个组

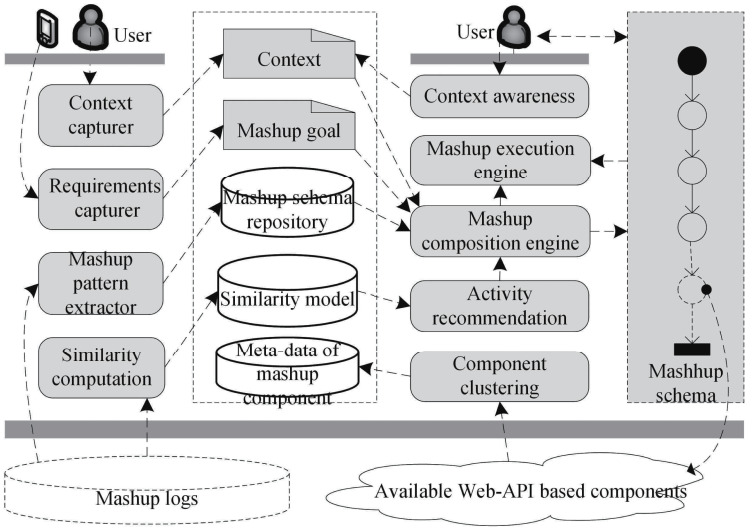

图 3 系统框架

Figure 3 System structure

件、在不同时间完成, 而非最后的一次性输出.

3 系统结构

图 3 描述了基于上下文和用户偏好的 Mashup 构造和服务推荐框架, 该框架主要由 4 部分组成: 模式挖掘与构建、用户输入与感知、Mashup 模式库以及 Mashup 构造与运行.

模式挖掘与构建部分实现为定期执行的任务, 包括 Mashup 模式抽取、用户行为相似度计算、Mashup 组件聚类处理. 历史日志记录了用户以往的 Mashup 活动信息, 包括活动执行顺序、执行活动的资源、上下文信息等, 从中可获得有价值的 Mashup 构造信息. Mashup 模式抽取基于历史日志中离散的执行实例识别出用户行为模式, 生成上下文相关的用户活动条件概率模型; 基于历史日志中大量用户参与的 Mashup 活动执行轨迹, 用户行为相似度计算模块负责计算特定上下文环境和部分活动轨迹的条件下, 用户对候选活动的偏好值, 进而计算用户之间的行为模式相似度; Mashup 组件聚类处理是基于可运行组件实体, 按功能性描述抽象为用户活动, 便于从大量离散的历史执行实例中发现用户行为规律.

用户输入与感知部分包括需求获取和上下文感知, 需求获取模块允许用户以自然方式描述需求, 然后将需求规范化为 mi.req 形式; 上下文感知负责收集和更新用户在 Mashup 构造和执行过程中的上下文.

Mashup 构造与运行是系统的核心部分, 包括 Mashup 构造引擎、执行引擎和活动推荐引擎. 基于用户需求、当前上下文和部分执行轨迹, 活动推荐引擎根据用户行为模式相似度矩阵, 计算当前用户对各候选活动的偏好值, 从而形成对用户下一步活动的推荐. 基于此, 构造引擎通过合并所推荐的候选活动对当前 Mashup 的构成进一步完善, 用户确认后由执行引擎绑定实际组件并执行, 执行结果作为部分执行轨迹反馈给活动推荐引擎, 作为推荐下一步活动的条件之一.

针对同一用户需求, 往往存在多个 Mashup 实现方案, 而由于用户专业知识、可获得信息等限制, 难以规划出一个最适合的、理想的方案, 上述框架基于对用户偏好及需求的挖掘, 通过自主构造并推

荐的候选活动选择方法, 为用户提供了一个更有效的 Mashup 自主构造的途径. 从下节开始, 讨论该系统中涉及的若干关键问题.

4 基于日志的 Mashup 模式挖掘

在定义 3 中, Mashup 实例表示一次独立的历史执行轨迹, 包括涉及到的组件及执行序列、执行过程中的上下文、相关用户信息等, 我们试图从大量孤立的 Mashup 实例中挖掘出若干种特定的用户行为模式, 作为对未来 Mashup 构建的支持.

定义4 (Mashup 活动) 用以描述特定功能的组件的抽象概念, 是对 Web-API 服务实体的抽象, 表示为元组: ma = ⟨Op, In, Out, Desc⟩, 其中 Op 表示操作名称, In = {in_1, in_2, ...} 和 Out = {out_1, out_2, ...} 分别是输入和输出参数集合, 每个参数表示为一个元组: p = ⟨name, Desc⟩, $p \in \text{In} \cup \text{Out}$. Desc 表示当前参数 / 组件标注关键字集合: Desc={anno_1, anno_2, ...}.

定义5 (Mashup 模式) 假定 A 是 Mashup 活动集合, C 是上下文集合, B 是 A 和 C 上的笛卡儿积: $B = A \times C$. B^* 表示 B 上的有限长度序列. 一个 Mashup 模式 σ 是 B^* 上的活动序列, $\sigma \in B^*$, 表示为: $\sigma = \langle (a_1, c_1), (a_2, c_2), \ldots, (a_n, c_n) \rangle$, $\sigma[i] = (a_i, c_i)$, $|\sigma| = n$ 是活动序列的长度, $1 \leqslant i \leqslant n$.

在以上定义中, 每个 "活动" 对应于多个满足其功能规格的 Web-API 服务实体, 于是, 一个 Mashup 模式本质上是多个具有相似上下文和行为特征的 Mashup 实例的抽象. 基于活动序列对包含大量离散 Mashup 实例的日志进行模式重建, 是一种简单、直观的途径, 因为任何一个具有复杂逻辑结构的活动执行模型总是可以由多个活动序列模型表达.

4.1 基于组件聚类的活动提取

作为构成 Mashup 模式的基本要素, 活动提取是模式挖掘的前提, 目前, 面向服务实体的分类或聚类方法已有较多研究, 本文采用一种基于功能性标注集合的语义聚类方法. 由于 Mashup 组件通常是基于轻量级的 Restful 协议实现, 而不需要像传统 Web Service 需要额外的 WSDL 描述文档, 本文假定已注册的 Web 组件通过基于 Html、Rdfa 标记语言的 SA-Rest 标注 [16] 进行描述. 对于任意两个服务 $\mathcal{A}$, $\mathcal{B}$, 其标注 $\mathcal{A}$.Desc, $\mathcal{B}$.Desc 分别表示为 AS={a_{11}, a_{12}, ..., a_{1m}}, BS={a_{21}, a_{22}, ..., a_{2n}}, 首先定义以下功能相似度度量函数:

$$\text{Sim}(\mathcal{A}, \mathcal{B}) = \left(|AS \cap BS| \times 2 + \sum_{i \in AS - BS} \sum_{j \in BS - AS} (1 - \text{dist}(i, j)) \right) / (m + n), \tag{2}$$

其中, dist(i, j) 表示术语 i, j 的语义距离.

除功能语义相似度之外, 同时考虑了接口兼容性, 即同一聚类中的组件应是可替换的, 即接口是相容的. 令 CompWith ($\mathcal{A}$,$\mathcal{B}$) 表示组件 $\mathcal{A}$ 的接口与 $\mathcal{B}$ 的兼容性, CompWith($\mathcal{A}$,$\mathcal{B}$) =1 表示组件 $\mathcal{A}$ 的接口与 $\mathcal{B}$ 兼容, 即组件 $\mathcal{B}$ 可被 $\mathcal{A}$ 替换:

$$\text{CompWith}(\mathcal{A}, \mathcal{B}) = \begin{cases} 1, & \sigma_{\text{In.Req='REQUIRED'}}(A) \subseteq \sigma_{\text{In.Req='REQUIRED'}}(B) \wedge (A.\text{Out} \cap B.\text{Out} \neq \varphi), \\ 0, & \text{else.} \end{cases} \tag{3}$$

于是, 基于功能相似度和接口兼容性度量函数, 对组件域中的 Web 组件进行聚类处理, 每个分类对应于一个 Mashup 活动, 使用 Clu($\mathcal{A}$) 表示组件 $\mathcal{A}$ 的所属分类:

$$\text{Clu}(\mathcal{A}) = \text{Clu}(\mathcal{B}) \Leftrightarrow \text{Sim}(\mathcal{A}, \mathcal{B}) >= \eta \quad \text{and} \quad (\text{CompWith}(\mathcal{A}, \mathcal{B}) = 1 \text{ or } \text{CompWith}(\mathcal{B}, \mathcal{A}) = 1), \tag{4}$$

表 1 用户活动轨迹偏好矩阵

Table 1 Matrix for user preference of activity trace

User/P	P_1	P_2	$\cdots$	P_n
U_1	$P_{1,1}$	$P_{1,2}$	$\cdots$	$P_{1,n}$
U_2	$P_{2,1}$	$P_{2,2}$	$\cdots$	$P_{2,n}$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
U_m	$P_{m,1}$	$P_{m,2}$	$\cdots$	$P_{m,n}$

其中, $0 \leqslant \eta \leqslant 1$ 是一个聚类影响因子, 用来调整组件相似性阈值和聚类规模.

4.2 Mashup 活动轨迹相似度

令 σ 表示历史日志中的一个 Mashup 实例, σ_t 表示 σ 的活动序列: σ_t=$\langle t_1, t_2, \ldots, t_n\rangle$, 活动序列 σ_t 中所有相邻活动 $\langle t_i, t_{i+1}\rangle$ $(1 \leqslant i \leqslant n-1)$ 构成的集合称为 σ 的活动邻接关系, 表示为 $\mathrm{ANR}(\sigma_t) = \{\langle t_i, t_{i+1}\rangle\}$, $1 \leqslant i \leqslant n-1$. 同时, 使用 $\mathrm{ADR}(\sigma_t)$ 表示 σ 的活动依赖关系: $\mathrm{ADR}(\sigma_t) = \{\langle t_i, t_j\rangle\}$, $i < j$, $1 \leqslant i \leqslant n-1$, $2 \leqslant i \leqslant n$. 可见, 活动邻接关系和活动依赖关系是对活动序列基于不同严格性的分解, 基于此, 定义活动轨迹相似度如下:

定义6 (活动轨迹相似度) 令 σ_1, σ_2 分别表示任意两个活动序列: σ_1=$\langle t_1, t_2, \ldots, t_n\rangle$, σ_2=$\langle t_1^{'}, t_2^{'}, \ldots, t_m^{'}\rangle$, σ_1, σ_2 的活动轨迹相似度定义为

$$\mathrm{Sim}_t(\sigma_1, \sigma_2) = \frac{|ANR(\sigma_1) \cap ANR(\sigma_2)|}{|ANR(\sigma_1) \cup ANR(\sigma_2)|} \times w + \frac{|ADR(\sigma_1) \cap ADR(\sigma_2)|}{|ADR(\sigma_1) \cup ADR(\sigma_2)|} \times (1-w), \tag{5}$$

其中, $0 \leqslant w \leqslant 1$, 用来调整活动邻接关系相似度和活动依赖关系相似度的权重.

4.3 用户行为模式相似度模型

具有相同或相似目标及行为特征的用户称为相似用户, 基于相似用户的协同过滤推荐方法已被多个领域证明其良好的有效性 [17], 本文研究小组曾提出一个面向流程片段的用户相似度模型 [18], 取得了较好的效果. 类似地, 在基于动态上下文的 Mashup 构造中引入用户偏好因素的影响, 无疑会提高 Mashup 活动推荐的质量. 由于用户选择的历史活动轨迹反映了该用户的偏好或其行为模式, 我们使用轨迹执行次数作为用户对活动轨迹的偏好值.

定义7 (用户活动轨迹偏好) 设 U 表示一个用户, 其活动轨迹偏好为一个二元组: EP=(P, δ), 其中, P 是用户 U 所执行活动轨迹种类的有限集合, $P = \{P_1, \ldots, P_n\}$. $\delta(P_i)$ 表示用户 U 对活动轨迹 P_i 的偏好值, 其值为历史日志中该用户执行活动轨迹 P_i 的次数.

表 1 为基于特定 Mashup 需求 R 的用户活动轨迹偏好矩阵 UP(R), 其中 P_1, P_2, $\ldots$, P_n 表示活动轨迹的不同聚类, U_1, U_2, $\ldots$, U_m 表示不同用户, $P_{i,j}$ 为基于日志统计的用户 U_i 活动轨迹偏好值 $\delta(P_j)$.

基于用户活动轨迹偏好矩阵, 可以度量任意用户之间的行为模式相似性. 在度量相似性方面, Pearson 相关系数 [17] 有很多优点, 不仅考虑了所有用户的平均偏好值, 还能有效识别用户对多种选择偏好差值的一致性, 本文利用 Pearson 相关度系数计算用户偏好指标的相似度:

$$r_{uv} = \frac{\sum_{i \in I_{uv}} (P_{u,i} - \overline{P_u}) \times (P_{v,i} - \overline{P_v})}{\sqrt{\sum_i \in I_{uv} (P_{u,i} - \overline{P_u})^2} \sqrt{\sum_i \in I_{uv} (P_{v,i} - \overline{P_v})^2}},$$

其中, r_{uv} 表示任意两个用户 u 和 v 的偏好相似度; I_{uv} 表示 u 和 v 共同的活动轨迹项集; P_{ui} 与 P_{vi} 为用户 u, v 对活动轨迹 P_i 的偏好值, P_{ui} 和 P_{vi} 表示 u, v 在 P_i 上的平均偏好值, $\overline{P_u}$ 和 $\overline{P_v}$ 表示 u, v 在 I_{uv} 上的平均偏好值.

定义8 (行为模式相似度) 设两个用户 U_i 与 U_j 的活动轨迹偏好分别是 EP_i=(P_i,δ_i) 和 EP_j=(P_j, δ_j), 其行为模式相似度定义为

$$A_{ui\Leftrightarrow uj}=\begin{cases} r_{ij}, & r_{ij\geqslant 0}, \\ 0, & \text{else}, \end{cases} \tag{6}$$

其中, r_{ij} 为两个用户在共同活动轨迹集合 $P_i \cap P_j$ 偏好值上的 Pearson 相关系数.

基于用户活动轨迹偏好矩阵和行为模式相似度度量式 (6), 对所有用户进行聚类处理, 所产生的分类集合为 CU=$\{\text{ro}_1, \text{ro}_2, \ldots, \text{ro}_m\}$, 使得 ro_i ($1 \leqslant i \leqslant m$) 中的任意两个用户 u_k, u_j, 满足

$$A_{u_k \Leftrightarrow u_j} \geqslant M, \quad \forall u_k \in \text{ro}_i, \ u_j \in \text{ro}_i,$$

即每个聚类中的用户划分为具有相似行为模式的用户, M 为条件因子, 可用来调节行为模式相似用户聚类的数量. 其中每个元素 ro_i, 也称之为角色.

用户聚类过程是离线完成的, 可对该矩阵进行周期性更新. 如同所有协同过滤方法, 需要解决 "冷启动" 问题. 我们采用的策略是直接对多个用户的活动偏好值求均值, 虽然在初始阶段降低了个性化活动推荐的质量, 但是该方法简单、有效. 随着用户偏好信息的增加, 所获得的推荐质量将逐渐提高.

4.4 基于日志的 Mashup 模式挖掘

假定 Web 组件经聚类处理后有 n 个分类, 使用 n 个 Mashup 活动表示: $\text{as}_1, \text{as}_2, \ldots, \text{as}_n$. 基于日志中离散的 Mashup 实例, 构造基于上下文的活动依赖矩阵, 用以映射用户在历史执行中的行为模式, 表 2 表示了针对某个特定 Mashup 需求和角色的矩阵 M(Re, Ro).

在上述矩阵中, as_i ($1 \leqslant i \leqslant n$), ct_j ($1 \leqslant j \leqslant m$) 分别表示活动和上下文, $T_{i,j,k}$ 表示在上下文 ct_k 中, 活动 as_i 完成后, 用户选择活动 as_j 的次数. 空活动 (null activity) 表示 Mashup 的开始或结束. 活动依赖矩阵的构造较为直观, 算法 1 简要描述了构造方法.

算法 1 基于上下文的 Mashup 活动依赖矩阵构造

Require: mashup 实例集合 Si, Mashup 需求 Re, 用户分类 Ro;
Ensure: 活动依赖矩阵 T; Initialize matrix T with 0 value for all the elements;
For each mashup instance l in Si with requirement Re and user cluster Ro
Let σ denote the execution trace of log l, σ=($\langle s_1, c_1\rangle$, $\langle s_2, c_2\rangle$, $\langle s_3, c_3\rangle$, …, $\langle s_n, c_n\rangle$);
$T_{0,k,q} = T_{0,k,q} + 1$, with as_k=Clu(s_1) and ct_q=c_1;
For i from 1 to $n-1$
$T_{j,k,q} = T_{j,k,q} + 1$, with $\text{as}_j = \text{Clu}(s_i)$, $\text{as}_k = \text{Clu}(s_{i+1})$ and $\text{ct}_q = c_i$;
End for
$T_{j,0,q} = T_{j,0,q} + 1$, with $\text{as}_j = \text{Clu}(s_n)$ and $\text{ct}_q = c_n$;
End for
Return T.

基于该矩阵, 对于特定上下文 ct_k 和某个活动 as_i, 用户选择活动 as_j 作为下一个活动的概率为

$$P(\text{as}_j|\text{as}_i, \text{ct}_k) = T_{ij,k} \Big/ \sum_{t=1}^{n} T_{i,t,k}, \tag{7}$$

表 2 基于上下文的活动依赖矩阵

Table 2 The context-based matrix for activity dependencies

		null	as_1	as_2	$\cdots$	as_n
null	ct_1	0	$T_{0,1,1}$	$T_{0,2,1}$	$\cdots$	$T_{0,n,1}$
null	ct_2	0	$T_{0,1,2}$	$T_{0,2,2}$	$\cdots$	$T_{0,n,2}$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
null	ct_m	0	$T_{0,1,m}$	$T_{0,2,m}$	$\cdots$	$T_{0,n,m}$
as_1	ct_1	$T_{1,0,1}$	0	$T_{1,2,1}$	$\cdots$	$T_{1,n,1}$
as_1	ct_2	$T_{1,0,1}$	0	$T_{1,2,2}$	$\cdots$	$T_{1,n,2}$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
as_1	ct_m	$T_{1,0,1}$	0	$T_{1,2,m}$	$\cdots$	$T_{1,n,m}$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
as_n	ct_1	$T_{n,0,1}$	$T_{n,1,1}$	$T_{n,2,1}$	$\cdots$	0
as_n	ct_2	$T_{n,0,2}$	$T_{n,1,2}$	$T_{n,2,2}$	$\cdots$	0
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
as_n	ct_m	$T_{n,0,m}$	$T_{n,1,m}$	$T_{n,2,m}$	$\cdots$	0

事实上, 在 mashup 活动执行过程中, 活动 as_j 不应仅依赖于其前驱活动 as_{j-1}, 而与前面用户已完成的 $j-1$ 个活动相关. 于是, 定义选择活动 as_j 的条件概率分布为

$$P(\mathrm{as}_j|\mathrm{as}_1,\mathrm{as}_2,\ldots,\mathrm{as}_i,\mathrm{ct}_k)=\begin{cases}T_{ij,k}/\sum_{t=1}^{n}T_{i,t,k}, & \mathrm{as}_j\notin\{\mathrm{as}_1,\mathrm{as}_2,\ldots,\mathrm{as}_i\},\\ 0, & \mathrm{as}_j\in\{\mathrm{as}_1,\mathrm{as}_2,\ldots,\mathrm{as}_i\}.\end{cases}\tag{8}$$

在上下文感知信息缺失的情况下, 可以基于所有上下文的统计信息进行计算:

$$P(\mathrm{as}_j|\mathrm{as}_1,\mathrm{as}_2,\ldots,\mathrm{as}_i)=\begin{cases}(\sum_{k=1}^{m}T_{i,j,k})/(\sum_{k=1}^{m}\sum_{t=1}^{n}T_{i,t,k}), & \mathrm{as}_j\notin\{\mathrm{as}_1,\mathrm{as}_2,\ldots,\mathrm{as}_i\},\\ 0, & \mathrm{as}_j\in\{\mathrm{as}_1,\mathrm{as}_2,\ldots,\mathrm{as}_i\}.\end{cases}\tag{9}$$

5 两阶段推荐算法

本节讨论 Mashup 自主构建方法, Mashup 的构建是一个目标逼近的迭代过程, 每次迭代包含两个阶段: Mashup 模式构建和 Mashup 实例执行. 针对用户需求、用户特征和当前上下文, 基于活动转移概率模型, 通过计算并推荐用户下一步的活动逐步构建出 Mashup 模式片段; 然后, 将当前活动绑定实际 Web 组件并执行. 重复以上过程, 直到用户需求获得满足. 以下讨论这一过程的核心部分, 即 Mashup 模式构造算法, 输入参数为用户、需求、上下文和已完成活动序列 (Mashup 片段), 算法计算并推荐下一个用户活动, 输出为一个增强的活动序列.

5.1 活动轨迹支持度

令 σ 表示一个 Mashup 实例的活动序列, σ_p 表示 σ 的前缀序列, ρ 是一个活动序列 (即 Mashup 片段), $\mathrm{Re}(\sigma)$ 和 $\mathrm{Re}(\rho)$ 分别表示 σ 和 ρ 的需求, 定义 Mashup 实例 σ 对活动轨迹 ρ 的支持函数, 即表示活动轨迹 ρ 演化为 Mashup 实例 σ 的概率:

$$S(\rho,\sigma)=\mathrm{Sim}_t(\rho,\sigma)\times w+S_g(\rho,\sigma)\times(1-w),\tag{10}$$

其中, $\mathrm{Sim}_t(\rho, \sigma)$ 是式 (5) 定义的活动轨相似度, 而 $S_g(\rho, \sigma)$ 表示 Mashup 需求的相似度:

$$S_g(\rho,\sigma)=\frac{|g(\sigma).\mathrm{Out}\cap g(\rho).\mathrm{Out}|}{|g(\sigma).\mathrm{Out}\cup g(\rho).\mathrm{Out}|}. \tag{11}$$

在构造活动依赖矩阵的过程中, 活动轨迹支持度函数用来过滤历史日志, 将与当前构造的 Mashup 实例无关的执行记录排除掉.

5.2 Mashup 构造算法

以下算法描述了 Mashup 迭代构造过程中模式构造阶段, 即候选活动推荐.

算法 2 是 Mashup 构造过程中每次迭代的模式构造阶段, 通过计算并推荐当前 Mashup 片段的下一个活动, 逐步增强 Mashup 片段, 直到满足用户需求. 上述算法中活动依赖矩阵的构造是一个耗时的任务, 其复杂度为 $O(n\times m^2)$, n 为 Mashup 日志中实例数量, m 为每个 Mashup 实例所含平均活动数量. 另外, 普适计算环境下的 Mashup 通常支持反馈和交互, 以带给用户良好的体验, 在这种情况下, 算法 RAMA 可修改为返回多个候选活动并由用户选择确认.

算法 2 基于 Mashup 执行轨迹的活动推荐 (RAMA)

Require: 已完成活动轨迹 f, mashup 需求 r, 当前用户 u, 轨迹支持度影响因子 h.
Ensure: 新的 mashup 片段 $\widehat{f}$.

令 f= ($\langle s_1, c_1\rangle$, $\langle s_2, c_2\rangle$, $\langle s_3, c_3\rangle$, ..., $\langle s_k, c_k\rangle$);
初始化 Mashup 实例集合 ss_i= {};
For (日志中的每个 Mashup 实例 m_i)
  利用式 (10) 计算轨迹支持度 $S(f, m_i)$;
  If ($S(f, m_i) \geqslant h$)
    $ss_i = ss_i \cup \{m_i\}$;
  End if
End for
基于 Mashup 实例集合 ss_i, 利用算法 1 构造活动依赖矩阵 $T(r, o)$; //$u\in o$ and $o\in CU$
na = null; $p(na)$=0; //设置推荐活动及其选择概率
For ($T(r, o)$ 中的每个活动 a)
  If (c_k is not null)
    $p(a)$ = P($\mathrm{as}_j|\mathrm{as}_1, \mathrm{as}_2, \ldots, \mathrm{as}_k, c_k$); // 基于式 (8);
  Else
    $p(a)$ = P($\mathrm{as}_j|\mathrm{as}_1, \mathrm{as}_2, \ldots, \mathrm{as}_k$); // 基于式 (9);
  End if
  If ($p(na)\leqslant p(a)$)
    $na = a$; $p(na) = p(a)$;
  End if
End for
$\widehat{f} = f \cup \langle na, \mathrm{null}\rangle$;
Return $\widehat{f}$.

5.3 基于上下文推理的算法优化

在实际应用场景中, 由于移动网络环境的不稳定性、动态性等原因导致用户上下文感知能力或上下文信息缺失的情况较为普遍, 在上述算法中, 根据上下文环境无关的活动流行度为用户推荐活动及

服务. 为提高上下文环境相关的推荐准确度, 本文提出一种基于用户活动序列的上下文推理方法, 对算法 RAMA 进行优化.

具体来说, 根据用户已完成的活动序列和上下文相关用户活动的条件概率分布, 利用 Bayes 公式, 计算当前活动或服务执行时用户上下文的概率分布. 即, 对于活动 ct_j, 用户所处上下文为 as_i 的概率为

$$P(\mathrm{ct}_j|\mathrm{as}_i) = P(\mathrm{as}_i|\mathrm{ct}_j) * P(\mathrm{ct}_j)/P(\mathrm{as}_i). \tag{12}$$

于是, 活动 as_i 所处的当前上下文 ct_i 可推理为具有最大条件概率的上下文 ct_j, 并且 ct_j 的条件概率满足 $\max_{1\leqslant j\leqslant m}P(\mathrm{ct}_j|\mathrm{as}_i)$, 即

$$\mathrm{ct}_i = \mathrm{ct}_j, \quad \text{with} \quad P(\mathrm{ct}_j|\mathrm{as}_i) = \max_{1\leqslant j\leqslant m}P(\mathrm{ct}_j|\mathrm{as}_i). \tag{13}$$

在算法 RAMA 中, 当 Mashup 日志中某个活动所关联的上下文缺失时, 即可采用上述基于 Bayes 模型的推理上下文, 即在算法中使用式 (13) 代替式 (11), 从而得到新的算法, 本文称之为算法 “RAMA with Context Inference”.

另外, 在算法 RAMA 中, 活动依赖矩阵的构造是一个耗时的任务, 其复杂度为 $O(n \times m^2)$, n 为 Mashup 日志中实例数量, m 为每个 Mashup 实例所含平均活动数量, 可以对算法进一步优化, 即活动依赖矩阵只在 Mashup 实例构造开始时计算一次, 此后每次迭代计算推荐活动时不再更新活动依赖矩阵, 性能优化后的算法称为 “Performance Improved RAMA”. 在新的算法中, 虽然活动依赖矩阵不再伴随活动推荐迭代过程实时更新, 对推荐精确度稍有影响, 但是可减少大量计算复杂度.

Mashup 模式构造阶段完成后, 下一个阶段是为推荐的活动绑定具体的 Web 组件并执行. 目前, 有大量研究工作关注 Mashup 服务推荐方法可供参考 [3~7]. 在本文方法中, 由于已注册的 Web 组件已基于语义相似度和 API 接口相容性聚类, 我们采用基于 QoS 的方法对上一阶段选择的活动绑定具体 Web 组件.

6 模拟实验

针对 Mashup 自主构造方法的性能和有效性, 本文构造数据进行了部分模拟试验, 本节简要介绍实验情况.

6.1 实验场景与实验数据构造

本节用一个为用户提供外出就餐相关服务的 Mashup 组合应用作为实验场景 (如图 4 所示). 在这一应用场景中, 用户的目标可描述为 “从当前位置出发到达 10 公里之内的某处餐馆就餐”, 从制定计划到就餐的过程, 用户可能经历多个不同的上下文环境, 期间涉及到的可用 Web 组件包括 LBS 位置服务、地图服务、导航服务、天气服务、预定服务、点餐服务、支付服务、评价服务等. 在基于特定场景的传统 Mashup 或服务组合方法中, 在初始阶段就需要用户明确过程中的各种活动, 并通过 Mashup 构造工具构造出完整的组合服务并立即执行. 事实上, 由于用户在这一过程中所处上下文环境的动态性和不可预知性, 没有足够的信息支持为用户预先定义出一个活动序列, 即使预先定义出一个组合服务, 其效果也未必理想. 在本文所设计的实验场景中, 以时间、地点、设备为主要属性构造出若干种不同的上下文环境, 通过引入用户活动时间点, 以及上下文环境随时间点的随机转换, 以尽可能反映用户活动场景的真实情况, 我们假定 Mashup 服务在帮助用户完成其目标的过程中, 用户所处的上下文

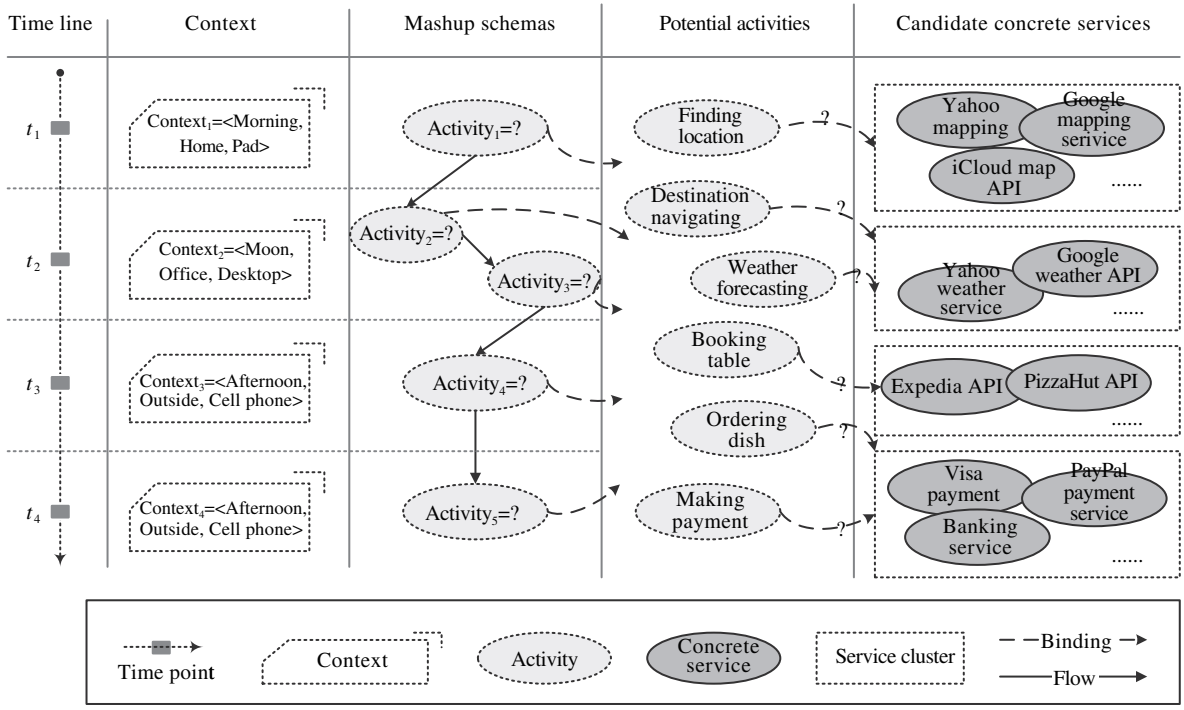

图 4　一个面向外出就餐 Mashup 服务场景

Figure 4　A scenario of mashup service for dining out

环境将要经历 4 次转换, 即 $\langle t_1, \text{Context}_1\rangle \rightarrow \langle t_2, \text{Context}_2\rangle \rightarrow \langle t_3, \text{Context}_3\rangle \rightarrow \langle t_4, \text{Context}_4\rangle$. 其中, t_i 和 Context_i $(1 \leqslant i \leqslant 4)$ 分别表示时间点和特定上下文. 并且, 在 t_i 时刻, Context_i 是可获取的, 而 t_{i+1} 及之后时刻的 Context 无法确定.

1. 上下文环境构造

本文以时间、地点、设备为主要属性构造用户活动所处的上下文环境, 这几种上下文属性在现实应用场景中也是可感知的. 各上下文属性取值范围如下所述.

时间属性: Time = Morning, Afternoon, Evening.

地点属性: Location = Indoor, Driving, Outside.

设备属性: Device = Desktop, MobileDevices.

根据上述 3 种属性的取值进行符合语义的组合, 形成多种上下文环境, 排除掉若干种属性值之间的互斥关系, 如 Location.Driving 与 Device.Desktop, Location.Outside 与 Device.Desktop 等, 所产生的上下文环境及相互转移关系如表 3 所示.

2. Web 组件构造

Web 组件是构成 Mashup 的基本单元, 目前 Internet 上有大量可用的服务资源, 主要两类: 基于 SOAP/Rest 协议的 Web 服务和面向浏览器的 Http 协议 Web Widget 组件, 在本文实验中统称为 Web 组件. 由于任何 Mashup 都是由语义或操作相关的 Web 组件构成, 主要面向上述实验场景的应用领域构造 Web 组件测试用例, 同时为验证本文方法中算法的性能和现实环境中的效果, 也增加了与本实验场景虽无直接关系, 但与其中某些操作具有语义关联的电子商务领域的 Web 组件. 通过构造组件注册库, 以关系数据库存储组件的元数据, 具体构造过程分为两个阶段: 首先, 对互联网上真实服务的搜索与抽取, 形成组件注册库中的原生服务, 服务来源包括通用 Web 服务提供商 (如百度地图

表 3 上下文环境及转换关系构造

Table 3 The contexts and their transitions

Contexts	Attribute values	Transferable contexts
Ctx_1	<Morning, Indoor, Desktop>	Ctx_3,Ctx_2,Ctx_4,Ctx_5,Ctx_6,Ctx_7,Ctx_8S
Ctx_2	<Morning, Indoor, MobileDevices>	Ctx_2,Ctx_3,Ctx_4,Ctx_5,Ctx_6,Ctx_7,Ctx_8
Ctx_3	<Morning, Driving, MobileDevices>	Ctx_1, Ctx_2,Ctx_4,Ctx_5,Ctx_6,Ctx_7,Ctx_8
Ctx_4	<Morning, Outside, MobileDevices>	Ctx_1, Ctx_2,Ctx_3,Ctx_5,Ctx_6,Ctx_7,Ctx_8
Ctx_5	<Afternoon, Indoor, Desktop>	Ctx_6, Ctx_7,Ctx_8,Ctx_9,Ctx_{10},Ctx_{11},Ctx_{12}
Ctx_6	<Afternoon, Indoor, MobileDevices>	Ctx_5, Ctx_7,Ctx_8,Ctx_9,Ctx_{10},Ctx_{11},Ctx_{12}
Ctx_7	<Afternoon, Driving, MobileDevices>	Ctx_5, Ctx_6,Ctx_8,Ctx_9,Ctx_{10},Ctx_{11},Ctx_{12}
Ctx_8	<Afternoon, Outside, MobileDevices>	Ctx_5, Ctx_6,Ctx_7,Ctx_9,Ctx_{10},Ctx_{11},Ctx_{12}
Ctx_9	<Evening, Indoor, Desktop>	Ctx_{10}, Ctx_{11},Ctx_{12}
Ctx_{10}	<Evening, Indoor, MobileDevices>	Ctx_9, Ctx_{11},Ctx_{12}
Ctx_{11}	<Evening, Driving, MobileDevices>	Ctx_9, Ctx_{10},Ctx_{12}
Ctx_{12}	<Evening, Outside, MobileDevices>	Ctx_9, Ctx_{10},Ctx_{11}

API, Mapquest Service, Google Place API, TeleNav GPS API 等), 和特定平台或商家提供的服务 (如 Dianping.com APIs, Yelp APIs, Ordr.in APIs, Amazon AWS 等), 主要抽取这些开放 Web-API 的接口描述和服务描述形成 Web 组件元数据, 然后根据其描述产生语义标注, 并随机产生性能、可靠性、易用性等 QoS 信息; 在第 2 阶段, 针对每个原生服务元数据, 生成一定范围内随机数量的虚拟 Web 组件, 以描述信息的形式存储在组件注册库中, 使得注册库中的组件达到一定数量. 表 4 描述了测试用 Web 组件数据的统计情况, 其中, Web 组件总数为 501, 基于功能相似度计算的聚类数量为 23, 每个 Web 组件分别标注 1, 3, 5 个关键词.

3. 历史执行轨迹构造

根据构造的 Web 组件描述数据、以及设定的 Mashup 需求和上下文环境及转换关系, 构造基于用户的历史执行轨迹, 亦即 Mashup 实例数据集, 作为本文方法挖掘 Mashup 模式并推荐新的个性化 Mashup 服务的基础. 首先采用随机选择组件的方式, 组件序列满足接口相容性约束; 然后, 从两个方面对 Mashup 实例数据进行调整: 一方面是增加相似用户执行轨迹的聚集性, 另一方面是对每个 Web 组件增加流行度和可靠性两个属性, 并使流行度属性对组件出现在 Mashup 实例中的次数有所影响, 从而使构造的日志更符合现实情况. 表 5 是 Mashup 历史执行轨迹数据的构造情况统计.

6.2 实验结果

首先验证本文方法中基于功能相似度和接口相容性的 Web 组件聚类, 每个实验构造的组件都由 3 组独立的标注进行描述, 3 组标注分别包含 1, 2, 3 个描述术语, 组件聚类中英文术语和中文术语之间的语义距离分别根据 Wordnet [2)]、同义词词林[3)] 计算. 图 5(a) 表示了在不同条件下的组件聚类, 结果表明只采用 1 个关键词术语的聚类效果最差. 另外, 如果相似度调整因子设置过高, 由于组件相似条件过于严格而使得聚类数量太大.

同时, 将本文基于功能相似度和接口兼容性的服务聚类方法 (简称 FSandIC-based clustering) 与

2) http://wordnet.princeton.edu/.

3) http://ir.hit.edu.cn/demo/ltp/Sharing_Plan.htm.

表 4 测试数据构造情况统计

Table 4 The statistics of testing data

Classifications of Web components	Web components	Actual services	Virtual services
Total	501	83	424
LBS-based services	25	4	21
Mapping services	36	6	30
Navigation services	18	3	15
Services for finding restaurants	59	9	50
Promotion services	36	6	30
Reservation services	32	5	27
Services for ordering meals	10	1	9
Payment services	24	4	20
Services for rating restaurants	42	7	35
Weather services	36	6	30
Trading services	18	3	15
Services for shopping guide	25	4	21
Logistics services	19	3	16
Ticketing services	30	5	25
.....	.....	.....	.....

表 5 Mashup 实例数据集构造情况统计

Table 5 The statistics of testing data for mashup instances

The data set for mashup instances	
Number of mashup instances	5000
Number of involved users	70
Number of involved components	473
Estimated number of involved activities	31
Number of mashup requirements	12
Number of contexts	6

若干同类方法进行了比较, 在文献 [19] 中, 作者分别提出了基于多维属性向量空间的欧式距离 (简称 ED-based clustering) 和余弦距离 (简称 CD-based clustering) 的服务相似度计算方法, 出于类似目的, 文献 [20] 提出了基于功能相似和过程相似的两种服务聚类方法, 由于该文中基于过程相似的聚类方法面向复合服务并且需要完备的过程模型, 实验仅选择与本文有可比性的基于功能相似的方法 (简称 FS-based clustering) 进行比较. 在 4 种方法的比较中, 采用了相同的语义词典用以计算术语之间的语义距离, 并将最终计算出所有服务之间的相似度归一化处理为 [0, 1] 区间的值, 其中方法 FS-based clustering 的服务基本描述及输入输出参数权重设置为 (ω_1=0.4, ω_2=0.3, ω_3=0.3). 图 5(b) 表示了 4 种方法在不同相似度调整参数下产生的聚类数量, 由实验结果可见, 本文采用的 FSandIC-based clustering 方法产生的聚类最多, 这是因为基于服务注解和接口相容性的聚类规则相比其他方法更为严格.

在另一个实验中, 基于 4 种服务聚类方法所产生的服务相似度矩阵, 对本文方法与另外 3 种方法

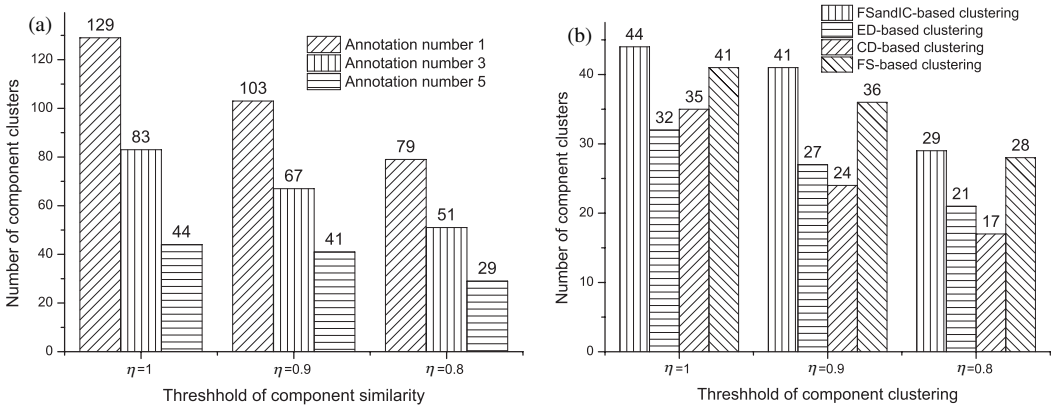

图 5 不同条件和方法的组件聚类结果比较

Figure 5 Component clustering with different conditions and approaches. (a) Clustering with different annotations; (b) Clustering with different approaches

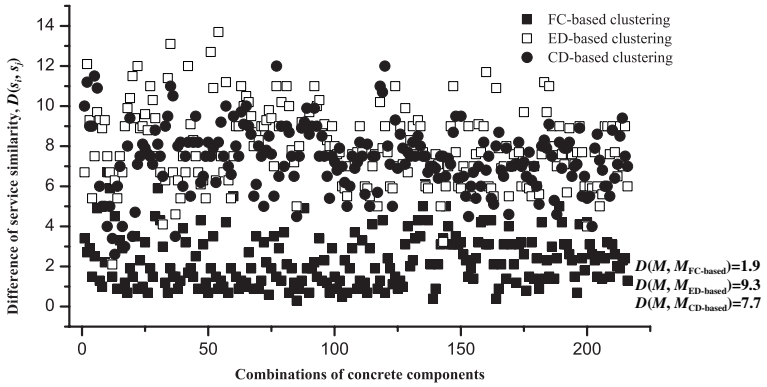

图 6 本文方法与其他方法的聚类差异度

Figure 6 Clustering differences among the approaches

分别计算其差异度, 包括服务对 (s_i,s_j) 的相似度差异值 $D(s_i,s_j)$=$d_1(s_i,s_j) - d_2\ (s_i,s_j)$, 以及总体差异度,

$$D(M_1,M_2)=\sqrt{\frac{\sum_{i=1}^{n}\sum_{j=i+1}^{n}(d_1(s_i,s_j)-d_2(s_i,s_j))^2}{n(n-1)/2}},$$

其中, $d_1(s_i,\ s_j\ )\in M_1$, $d_2(s_i,\ s_j)\in M_2$ 分别表示服务 s1, s2 在矩阵 M_1, M_2 中的相似度计算值. 图 6 显示了本文采用的 FSandIC-based clustering 方法与其他 3 种方法的聚类差异度, 由图中可见, 本文方法与 FC-based 聚类方法所产生的结果差异性较小, 这是因为两种方法都同时考虑了服务的功能性语义描述和输入输出参数, 而另外两种方法侧重于功能语义、领域信息、位置信息、QoS 等多维度空间, 并未考虑输入输出参数的匹配或相容性问题.

其次, 通过实验验证 Mashup 构造的有效性. 在此实验中, 将 Mashup 实例分为两部分, 训练数据集和基准测试数据集, 实验基于训练数据集构造 Mashup 模式并推荐 Web 组件, 然后将结果与基准测试数据集中相似需求和用户的 Mashup 实例进行比较. 下面定义对于特定 Mashup 目标 g 和用户 u,

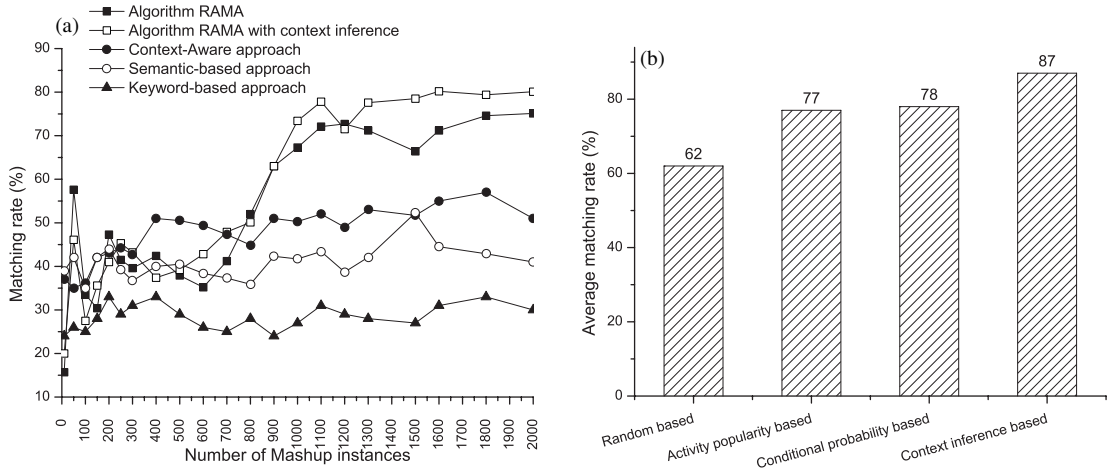

图 7 Mashup 实例构造匹配率

Figure 7 Matching rate of mashup constructions. (a) Matching rate of different approaches; (b) average matching rate of different approaches

相对于基准测试数据集的匹配率:

$$mr(g,u)=\frac{\sum_{c_i\in C}|S_b(g,r,c_i)\cap S_r(g,r,c_i)|}{\sum_{c_i\in C}|S_b(g,r,c_i)|},$$

其中, $S_r(g,r,c)$, $S_b(g,r,c)$ 分别表示本文算法构造的和基准测试数据集中的基于角色 r、上下文 c 的 Mashup 实例所含 Web 组件集合, $|S_b\ (g,r,c_i)\cap S_r\ (g,r,c_i)|$ 表示 Mashup 实例集合中所含共同 Web 组件的数量. 基于以上匹配率定义, 针对不同 Mashup 实例构造数量对本文 RAMA 活动推荐迭代算法和其他方法进行了比较, 我们研究了服务组合领域有代表性的方法, 包括上下文感知的服务组合方法 [9,10], 普适环境下的服务组合 [11,12], 基于规划的组合方法 [13,14,21,22] 等, 尽管这些方法的应用场景和模式与本文有所差别, 但是其中组合服务逻辑自动生成方法与本文 Mashup 服务自主构造方法有一定可比性, 试验中对这些方法简化, 仅提取其核心思想予以实现, 并与本文方法的构造结果进行比较, 如图 7(a) 所示. 随着用以评估的 Mashup 实例构造数量的增长, 本文方法产生的结果与基准测试数据集的匹配率逐渐趋于稳定. 对于另外的方法, 上下文感知的方法匹配率高于其他方法, 这是因为为了更自然的反应现实情况, 在测试数据的构造中考虑了上下文变化, 用户偏好等因素的影响, 而基于语义和关键词的组合方法仅侧重于考虑服务 QoS、功能匹配度等方面. 图 7(b) 表示另一个实验中不同方法的匹配率. 由实验结果可见, 对于缺乏上下文信息的情况, 基于 Bayes 公式的上下文推理方法可以推测出较高概率的上下文环境, 从而提高服务推荐质量.

另一个实验针对用户活动序列的构造或推荐效果进行验证, 即同时检查用户活动以及活动之间的依赖关系, 实验基于用户活动轨迹偏好矩阵 (如表 1 所示) 和用户行为序列相似度, 通过对活动序列的评分预测进行验证, 将用户活动轨迹偏好矩阵中的轨迹选择次数归一化处理为 0~100 之间的评分, 基于变换后的评分矩阵, 在给定前驱活动序列 P_i 的前提下, 计算整体活动序列推荐的 MAE, 如图 8 所示, 本文考虑用户行为模式的迭代式推荐方法, 对活动序列的预测优于常规上下文感知的 CF 方法, 这是因为其他方法关注于单一活动或服务的推荐, 未考虑用户活动的过程性和依赖性.

最后, 我们对 Mashup 构造算法的性能也进行了评估, 在算法 2 的每次迭代, 基于上下文和相似用户的活动依赖矩阵都被重新构造一次, 显然这种策略的计算复杂度过大. 为提高性能, 我们在每次构

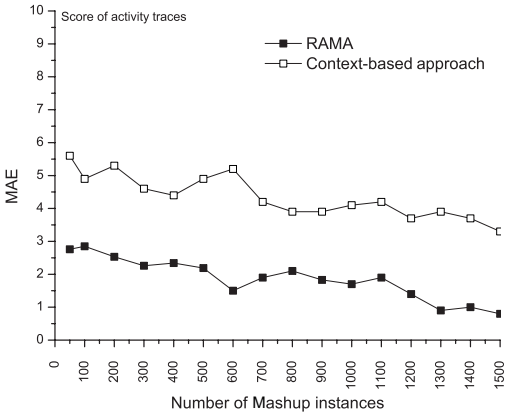

图 8 用户行为序列预测实验

Figure 8 Prediction results for activity traces

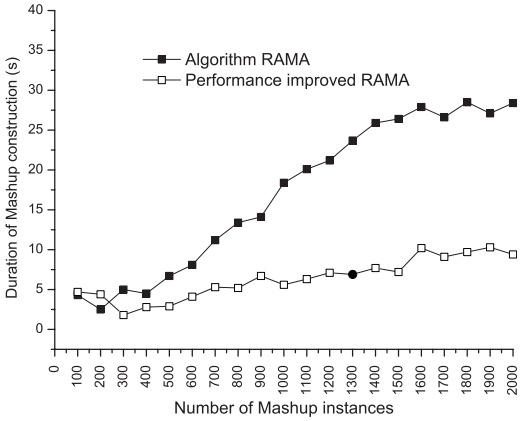

图 9 Mashup 模式构造算法性能

Figure 9 Performance for mashup construction

造一个 Mashup 实例时仅构造一次活动依赖矩阵, 而不是每次迭代时都要重构. 图 9 表明了性能改进后的算法在性能和推荐质量之间的折中.

6.3 实验结果讨论

针对移动计算环境中上下文的动态不确定性和普适性, 目前面向普通用户的推荐系统局限于特定上下文中离散服务的选择; 在组合服务推荐方面, 目前提出的 Mashup 服务需要用户自行规划详细的服务组合流程并设定执行时机, 而各种基于规划的自动服务组合方法也是根据预先设定的条件一次性生成一个服务组合方案, 在面向普通用户的移动计算环境下并不适用, 或者难以取得良好的效果. 本文提出的 Mashup 迭代式构造方法采用一种新的服务组合模式, 将 Mashup 构造与运行融入到用户行为过程, 即 "执行 – 构造" 的迭代过程, 使得下一步活动的选择能够获得更多的可用信息, 从而获得较优的推荐效果. 值得注意的是, 虽然本文方法通过扩展协同过滤算法使得推荐结果在满足用户偏好方面能够获得良好的效果, 但并不能保证其全局最优性, 因为在无数可能的组合方案中寻找全局最优的个性化结果无论在理论上, 还是在现实应用场景中并不可行. 但是, 由于在每个活动 a_i 的选择推荐时刻 t_{i-1}, 本文方法总是能够获得最多的可用信息, 包括即时上下文, 用户前期行为序列等, 这对于下一步活动的预测和推荐至关重要, 但是这些信息在 Mashup 构造的初始时刻 t_0 是无法获得的, 因此, 面对移动环境上下文动态性和普适性特点, 本文方法在用户行为模式预测和活动推荐方面能够获得比其他服务组合方法更好的效果, 上述 Mashup 构造的有效性实验结果也说明了这一点.

7 相关工作

当前针对 Mashup 技术的研究集中于面向非专业的普通用户提供快速、直观的开发方法 [1], 研究内容主要关注 Mashup 服务选择与推荐, 即从大量无组织和缺乏足够语义信息的候选组件中为特定用户选择并推荐合适的 Mashup 服务 [2], 典型方法包括采用基于功能语义的匹配方法发现符合功能性要求的组件 [4,5], 基于组件 QoS 选择与推荐 [3,23], 基于协同过滤方法 [6] 以及基于社交关系的面向用户的 Web 组件推荐 [7,8] 等. 这些组件推荐方法关注于对构成 Mashup 的特定组件进行选择与推荐, 其前提

是已经预先规划了整体 Mashup 模式, 普遍未考虑 Mashup 活动之间的关系对服务选择的影响, 同时本文所关注的 Mashup 模式的构建对用户和 Mashup 工具来说同样是一个挑战.

近年来, 上下文感知推荐方法已成为推荐系统最为活跃的研究内容之一 [15, 24], 随着移动互联网的普及, 用户在不同的上下文环境中 (网络状况、终端设备、时间、地点等) 表现出不同的行为模式和活动偏好, 基于上下文的服务推荐引起研究者的关注, 例如有研究者根据大量用户在不同物理场景下的活动选择历史, 基于用户进行角色挖掘 [25]. 文献 [26] 的作者将用户的服务需求分解为功能性和适应性场景, 使用后者用来应对上下文环境的动态变化. 为提高推荐质量, 本文对上下文概念范畴进一步扩展, 除用户环境信息外, 还包括用户目标和前期活动序列.

由于在电子商务等领域取得的良好效果, 个性化推荐技术被广泛应用于服务发现和推荐方法研究中. 文献 [27] 使用关键词标识用户的偏好, 并采用基于相似用户的协同过滤方法推荐个性化服务. 文献 [23] 采用一种推荐可视化技术提高推荐的可理解性和推荐结果的准确性, 作者工作侧重于服务 QoS, 同时基于协同过滤算法实现个性化推荐. 为改进推荐质量, 近年来有研究者将上下文因素和协同过滤方法结合起来考虑, 文献 [28] 基于用户对目标的偏好在不同场景中可能发生变化的考虑, 提出一种聚合用户上下文的协同过滤推荐方法. 文献 [29] 将 “用户 – 项目” 评分矩阵扩展为 “用户 – 项目 – 上下文” 多维度矩阵, 从而支持上下文感知的推荐. 而文献 [30] 的作者关注于上下文信息对不同用户权重差异性, 通过预测用户特定上下文评分进行个性化推荐. 基于类似的动机, 文献 [31] 通过引入上下文对协同过滤方法进行扩展, 预测用户在不同场景下的行为并推荐相应的活动. 目前的研究方法基本是在传统 “用户 – 项目” 评分矩阵和用户相似度模型基础上, 将 “项目” 简单映射为服务或用户活动, 然而, 这类模型并无法精确反映用户在行为模式上的偏好.

此外, 针对互联网上大量, 无序的服务进行分类或聚类处理, 是服务推荐方法的关键前提. 在服务语义模型方面, 文献 [32] 提出一种基于功能语义等的多维度 Web 服务语义社会标注方法. 文献 [33] 根据用户请求对服务库中的服务进行层次划分, 最终获得较为纯净的服务源信息. 文献 [34] 利用复杂网络社区发现算法对软件网络进行聚类, 实现服务的自动分类, 进而基于服务可组合关系及使用场景提出了相应的服务推荐算法.

8 总结

用户上下文的动态不确定性和普适性是移动计算环境的主要特征, 在这样的环境下, 面向普通用户的 Mashup 服务具有过程复杂性、时空复杂性、上下文感知复杂性等特点, Mashup 服务需要随时感知用户所处场景及其变化, 为用户即时规划合适的组合服务. 目前提出的 Mashup 服务需要用户自行规划详细的服务组合流程并设定执行时机, 而各种基于规划的自动服务组合方法也是根据预先设定的条件一次性生成一个服务组合方案, 在面向普通用户的移动计算环境下并不适用, 或者难以取得良好的效果. 基于此, 本文对 Mashup 服务研究的关注点从离散服务推荐转移到伴随用户行为的 Mashup 构造与运行过程, 提出一种新的基于迭代式自主构造的 Mashup 服务组合模式, 主要思想是对大量离散、孤立的历史执行轨迹进行挖掘处理, 从中发现潜在的基于上下文和用户特征的行为序列模式, 为用户构造优化的 Mashup 组合方案并推荐合适的 Web 组件提供支持. 分析和模拟实验表明, 本文方法能够有效简化 Mashup 构造过程中用户参与的复杂性, 减少对用户专业知识的依赖, 同时提高面向动态上下文和用户行为偏好的 Mashup 服务构造质量.

正如所有基于协同过滤的个性化推荐方法一样, 冷启动和稀疏矩阵是两个固有的缺陷, 本文采用基于平均偏好值的推荐解决用户选择和相似用户数据不足的问题, 但是在个性化推荐质量方面并不能

令人满意, 这也是未来工作的一个方向.

参考文献

1 Elmeleegy H, Ivan A, Akkiraju R, et al. Mashup advisor: a recommendation tool for mashup development. In: Proceedings of the IEEE International Conference on Web Services. New York: IEEE, 2008. 337–344

2 Greenshpan O, Milo T, Polyzotis N. Autocompletion for mashups. In: Proceedings of the VLDB Endowment. Lyon: ACM, 2009. 538–549

3 Picozzi M, Rodolfi M, Cappiello C, et al. Quality-based recommendations for Mashup composition. In: Proceedings of the 10th International Conference on Web Engineering. Berlin: Springer, 2010. 360–371

4 Bianchini D, De Antonellis V, Melchiori M. Semantics-enriched web APIs selection for enterprise Mashup development. In: Information Systems: Crossroads for Organization, Management, Accounting and Engineering. Berlin: Springer, 2012. 95–103

5 Ngu A H H, Carlson M P, Sheng Q Z, et al. Semantic-based mashup of composite applications. IEEE Trans Serv Comput, 2010, 3: 2–15

6 Zheng Z, Ma H, Lyu M R, et al. Qos-aware web service recommendation by collaborative filtering. IEEE Trans Serv Comput, 2011, 4: 140–152

7 Maaradji A, Hacid H, Skraba R, et al. Social-based web services discovery and composition for step-by-step Mashup completion. In: Proceedings of the IEEE International Conference on Web Services. New York: IEEE, 2011. 700–701

8 Cao B, Liu J, Tang M, et al. Mashup service recommendation based on user interest and social network. In: Proceedings of the IEEE International Conference on Web Services. New York: IEEE, 2013. 99–106

9 Medjahed B, Atif Y. Context-based matching for web service composition. Distrib Parall Databases, 2007, 21: 5–37

10 Beltran V, Arabshian K, Schulzrinne H. Ontology-based user-defined rules and context-aware service composition system. In: Proceedings of the Semantic Web: ESWC Workshops. Berlin: Springer, 2012. 139–155

11 Zhou J, Gilman E, Palola J, et al. Context-aware pervasive service composition and its implementation. Pers Ubiquit Comput, 2011, 15: 291–303

12 Mokhtar S, Fournier D, Georgantas N, et al. Context-aware service composition in pervasive computing environments. In: Proceedings of the 2nd International Workshop, RISE 2005. Berlin: Springer, 2006. 129–144

13 Niu W, Li G, Tang H, et al. CARSA: a context-aware reasoning-based service agent model for AI planning of web service composition. J Netw Comput Appl, 2011, 34: 1757–1770

14 Sirin E, Parsia B, Wu D, et al. HTN planning for web service composition using SHOP2. J Web Semant, 2004, 1: 377–396

15 Adomavicius G, Tuzhilin A. Context-aware recommender systems. In: Recommender Systems Handbook. Berlin: Springer, 2011. 217–253

16 Sheth A P, Gomadam K, Lathem J. SA-REST: semantically interoperable and easier-to-use services and Mashups. IEEE Internet Comput, 2007, 11: 91–94

17 Good N, Schafer J B, Konstan J A, et al. Combining collaborative filtering with personal agents for better recommendations. In: Proceedings of the 16th National Conference on Artificial Intelligence. Menlo Park: AAAI, 1999. 439–446

18 Li T, He W, Cui L Z, et al. Autonomous constructor of process based on personalized characteristics and preferences. Comput Integr Manuf Syst, 2013, 19: 1906–1912 [李婷, 何伟, 崔立真, 等. 基于用户偏好的个性化流程自主构建方法. 计算机集成制造系统, 2013, 19: 1906–1912]

19 Platzer C, Rosenberg F, Dustdar S. Web service clustering using multidimensional angles as proximity measures. ACM Trans Internet Tech, 2009, 9: 1–26

20 Sun P, Jiang C J. Using service clustering to facilitate process-oriented semantic web service discovery. Chinese J Comput, 2008, 31: 1340–1353 [孙萍, 蒋昌俊. 利用服务聚类优化面向过程模型的语义 Web 服务发现. 计算机学报, 2008, 31: 1340–1353]

21 Xu M, Cui L Z, Li Q Z. An extended graph-planning based Top-K service composition method. Acta Electron Sinica, 2012, 40: 1404–1409 [徐猛, 崔立真, 李庆忠. 基于扩展图规划的 Top-K 服务组合方法研究. 电子学报, 2012, 40: 1404–1409]

22 Hatzi O, Vrakas D, Nikolaidou M, et al. An integrated approach to automated semantic web service composition through planning. IEEE Trans Serv Comput, 2012, 5: 319–332
23 Chen X, Zheng Z, Liu X, et al. Personalized QoS-aware web service recommendation and visualization. IEEE Trans Serv Comput, 2013, 6: 35–47
24 Truong H L, Dustdar S. A survey on context-aware web service systems. Int J Web Inf Syst, 2009, 5: 5–31
25 Wang J, Zeng C, He C, et al. Context-aware role mining for mobile service recommendation. In: Proceedings of the 27th Annual ACM Symposium on Applied Computing. New York: ACM, 2012. 173–178
26 Hussein M, Han J, Yu J, et al. Scenario-based validation of requirements for context-aware adaptive services. In: Proceedings of the IEEE International Conference on Web Services. New York: IEEE, 2013. 348–355
27 Meng S, Dou W, Zhang X, et al. KASR: a keyword-aware service recommendation method on MapReduce for big data application. IEEE Trans Parall Distrib Syst, 2014, 25: 3221–3231
28 Shin D, Lee J, Yeon J, et al. Context-aware recommendation by aggregating user context. In: Proceedings of the IEEE Conference on Commerce and Enterprise Computing. New York: IEEE, 2009. 423–430
29 Karatzoglou A, Amatriain X, Baltrunas L, et al. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In: Proceedings of the 4th ACM Conference on Recommender Systems. New York: ACM, 2010. 79–86
30 Gao M, Wu Z. Personalized context-aware collaborative filtering based on neural network and slope one. In: Proceedings of the International Conference on Cooperative Design, Visualization, and Engineering. Berlin: Springer, 2009. 109–116
31 Chen A. Context-aware collaborative filtering system: predicting the user's preference in the ubiquitous computing environment. In: Proceedings of the 1st International Workshop on Location- and Context-Awareness. Berlin: Springer, 2005. 244–253
32 Ning D, He K Q, Peng R, et al. An automatic emerging approach for web service semantics based on social tagging. Chinese J Comput, 2011, 34: 2414–2426 [宁达, 何克清, 彭蓉, 等. 基于社会标注的 Web 服务语义自动浮现方法. 计算机学报, 2011, 34: 2414–2426]
33 Deng S G, Huang L T, Wu B, et al. QoS optimal automatic composition of semantic web services. Chinese J Comput, 2013, 36: 1015–1030 [邓水光, 黄龙涛, 吴斌, 等. 一种 QoS 最优的语义 Web 服务自动组合方法. 计算机学报, 2013, 36: 1015–1030]
34 Pan W F, Li B, Shao B, et al. Service classification and recommendation based on software networks. Chinese J Comput, 2011, 34: 2355–2369 [潘伟丰, 李兵, 邵波, 等. 基于软件网络的服务自动分类和推荐方法研究. 计算机学报, 2011, 34: 2355–2369]

A new paradigm for personalized mashup recommendation based on dynamic contexts in mobile computing environments

Wei HE[1,2*], Lizhen CUI[1,2], Guozhen REN[1], Qingzhong LI[1,2] & Ting LI[1]

1 *College of Computer Science and Technology, Shandong University, Ji'nan* 250101*, China*;
2 *Shandong Key Laboratory of Software Engineering, Shandong University, Ji'nan* 250101*, China*
*E-mail: hewei@sdu.edu.cn

Abstract With the rapid development of mobile Internet and increasing number of intelligent terminals, Internet services are now integrated into people's daily lives. In mobile computing environments, challenges have emerged for unprofessional user oriented mashup servers with new features, such as dynamic contexts and user preference. In this paper, we focus on the entire construction process rather than individual service recommendation for mashups, and propose a context-based and user preference-based autonomous construction approach for mashups. The main idea is to construct a context- and preference-related conditional probability model for user behavior patterns based on pattern mining of historical mashup logs, which can provide support for the generation of optimal personalized mashup solutions and also recommend suitable web components. Analysis and

simulation experiments conducted indicate that the proposed approach can autonomously generate personalized mashups with quality components without dependence on user professionalism.

Keywords mobile computing, context, mashup, service recommendation, collaborative filtering

Wei HE was born in 1972. He received a Ph.D. degree from the School of Computer Science and Technology, Shandong University, Jinan, in 2009. Currently, he is an associate professor in the School of Computer Science and Technology at Shandong University. His research interests include service computing, business process management, and recommendation algorithms.

Guozhen REN was born in 1970. He received a Ph.D. degree from the School of Computer Science and Technology, Shandong University, Jinan, in 2011. Currently, he is an associate professor in the School of Computer Science and Technology at Shandong University. His research interests include network security, mobile computing, and Internet of things.

Lizhen CUI was born in 1976. He received a Ph.D. degree from Shandong University, Jinan, in 2006. He is currently a professor in the School of Computer Science and Technology at Shandong University. His research interests include service computing, workflow and distributed data management for cloud computing, and service computing.

Qingzhong LI was born in 1965. He received a Ph.D. degree from the Institute of Computing Technology, Chinese Academy of Sciences in 2000. He is currently a professor and doctoral supervisor at Shandong University. His research interests are web information integration, artificial intelligence, and large-scale network data management.