基于频繁子图挖掘的数据服务Mashup推荐

张仲妹[1,2]，王桂玲[2]，张 赛[2]，韩燕波[2]

(1. 天津大学计算机科学与技术学院 天津 南开区 300072；2. 北方工业大学云计算研究中心 北京 石景山区 100144)

【摘要】该文基于大量用户的数据服务Mashup方案历史记录，通过将其构造成图并进行频繁子图挖掘，形成包含数据服务关联关系的知识库，提出一种能够根据用户选取的数据服务推荐后续数据服务列表的推荐算法。在构建知识库的过程中考虑数据服务后续的一元操作，提出了数据操作单元的概念，方便分析数据服务之间的关联关系，并采用了gSpan算法来挖掘数据服务的频繁子集。最后通过Yahoo! Pipes以及模拟数据集验证，该方法能够在用户创建数据服务Mashup的过程中有效快速地推荐数据服务。

关 键 词 数据服务； 频繁子集； gSpan算法； Mashup； 推荐系统
中图分类号 TP393 文献标志码 A doi:10.3969/j.issn.1001-0548.2016.03.018

Data Service Mashup Recommendation Method Based on Frequent Sub-Graph Mining

ZHANG Zhong-mei[1, 2], WANG Gui-ling[2], ZHANG Sai[2], and HAN Yan-bo[2]

(1. School of Computer Science and Technology, Tianjin University Nankai Tianjin 300072;
2. Department of Research Center for Cloud Computing, North China University of Technology Shijingshan Beijing 100144)

Abstract Since the number of data services is large and the user requirements are often unclear, users with no or little professional programming knowledge may spend much time but still can't build satisfying Mashup. We build a repository including the relationships of data services based on large number of the existing data service Mashup plans. The relationships are retrieved by constructing the Mashup plans into a set of graphs and mining the frequent sub-graphs from them. We propose a recommendation method to recommend a list of the subsequent data services according to some previous data services. In consideration of the subsequent operations following the previous data services when analyzing the relationships between two connected data services, a concept called data operate unit (DOU) is proposed to represent the data service and associated operations. Experiments based on Yahoo! Pipes datasets and simulated datasets are carried out. The experiments results show the proposed method can recommend data services effectively and efficiently while building data service Mashups.

Key words data as a service; frequent subset; gSpan; Mashup; recommendation system

数据Mashup是一种特定的Mashup[1]，允许用户从不同的数据源获得数据、处理数据、组合数据并以此来解决一类数据集成问题[2-4]。为了解决数据类型复杂的问题，许多商业机构将数据以数据服务[5-7]的形式提供给用户。有很多机构甚至提供了web Mashup平台，如Yahoo! Pipes[8]和Damia[9]以及DSS (date service space)平台[10]，使用户不需要编写程序，仅通过简单地拖拽和菜单操作就可以完成数据Mashup，本文中称这种数据Mashup为数据服务Mashup。

对非专业用户来说，形成符合期望的数据Mashup方案仍面临许多问题。用户面对众多的数据服务，需要花费大量时间构建数据服务Mashup，并且由于需求不明确，最终得不到满意的数据服务Mashup方案。如在ProgrammableWeb.com[11]上就有多达11 446个API供用户选择, Yahoo! Pipes需要用户自己寻找数据源。为了解决这些问题，本文提出一种数据服务推荐方法，在用户给出部分数据服务Mashup后，为用户推荐后续数据服务列表。该方法主要基于群体智慧, 通过对已有的数据Mashup方案进行预处理、挖掘频繁子图并分析得到储存数据服务关联关系的知识库，利用该知识库来完成推荐

收稿日期：2014－10－14； 修回日期：2015－02－27
基金项目：北京市自然科学基金(4131001)；北京市教育委员会科技计划面上项目(KM201310009003)；北京市属高等学校创新团队建设与教师职业发展计划项目(IDHT20130502)
作者简介：张仲妹(1990－)，女，博士生，主要从事服务计算方面的研究.

方法。

本文的主要贡献有：1) 提出数据操作单元的概念，用数据操作单元之间的关联关系来反映数据服务之间的关联关系；2) 将已有的数据服务Mashup方案构造成混合图并通过挖掘频繁子集分析生成知识库；3) 提出基于该知识库的推荐算法，在用户给出部分数据Mashup后，为用户推荐相关的、用户可能感兴趣的数据服务列表。

1 相关工作

对于毫无经验的用户来说，构造一个数据服务Mashup来满足自身的需求具有一定的难度。许多学者针对这一问题进行了关于Mashup推荐的研究。如文献[12-13]认为用户在构建Mashup时会共享一些一般特征，具有相同特征的用户往往会选择相似的组件，使用相似的组合模式并利用历史记录来计算数据服务的重要度，利用继承性计算当前的数据服务以及推荐的组合模式的距离来进行推荐。Mashup Advisor[12]利用知识库计算输出的流行度，再根据Mashup的当前状态推荐特定的输出，当选择好了一个输出后，系统会根据语义匹配修改当前Mashup所选择的输出。多数的研究工作在分析数据服务之间的关联关系时，并没有考虑一元操作对数据服务Mashup方案的影响。本文考虑数据服务之后的一元操作，提出了数据操作单元的概念，利用数据操作单元来代表对应的数据服务。

此外，很多学者利用模拟数据集来测试各自推荐算法的效率, 如文献[12]分析了ProgrammableWeb.com中的Mashup和API的分布特征，利用这些特征生成了模拟数据集。文献[13]按照自定义的规则生成模拟数据集，如数据服务在Mashup中随机分布。本文参考文献[12]并引入数据服务分类的特征生成模拟数据集测试推荐算法的效率。文献[14]中提出利用等价代换来优化Mashup的结构，提高Mashup的执行效率。

本文重点不是提高Mashup的执行效率，而是尽量保证数据服务Mashup方案的完整性。文献[15]将所有的Mashup合并构造成一张有向有权图，通过对这张有向有权图进行宽度优先遍历寻找权值超过一定阈值的有向边进行频繁子图的挖掘来辅助进行推荐。本文中Mashup构成的图为混合图，使用gSpan算法进行频繁子集的挖掘。

2 实例与问题定义

本节描述一个数据服务Mashup实例，并定义了用户在进行数据服务Mashup时可能遇到的问题。在该实例中，某电子商务公司的分析人员为了分析顾客在本季度的购买行为，以便为顾客进行针对性的服务，选择了几个相关的数据服务形成一个数据服务Mashup的方案，最终得到了顾客在本季度内的购买信息。该公司的数据服务如表1所示，构建的数据服务Mashup如图1所示。

表1 数据服务的详细信息

名称	ID	描述	输入	输出
getCustomerId	CID	根据顾客姓名获得顾客id	name	Id
getCustomer Detail	CD	根据顾客姓名获得顾客详细信息	name	address, age
getOrderInfo	OI	根据顾客id获得顾客的订单信息	id	ordered, productId, orderNumber
getReturnInfo	RI	根据顾客id获得顾客退货信息	id	ordered, returned, returnNum, reason
getChangeInfo	CHI	根据顾客id获得顾客的换货信息	id	ordered, changed, changeNum, resaon
getCustomer Comment	CC	根据顾客id获得顾客的评价	id	ordered, comment
⋮	⋮	⋮	⋮	⋮

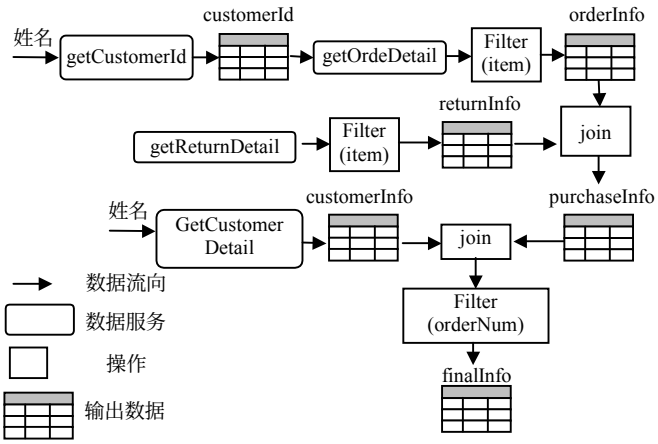


图1 数据服务Mashup方案实例

在进行数据服务Mashup时可能会遇到以下问题：

1) 对众多的数据服务没有系统了解，在选择数据服务时需要花费较多的时间查看数据服务的详细信息，降低形成数据服务Mashup方案的效率。

2) 对公司需求理解不够全面，会遗漏一些相关的数据服务，如数据服务getCustomerComment、getChangeInfo。而这些数据服务所提供的信息也可以帮助公司分析顾客的购买行为。

3) 分析人员可能会忽略更合适的数据服务而选择自己更了解的数据服务，增加数据服务Mashup方案的复杂度。

本文针对问题1)、2)提出了基于群体智慧的数据服务Mashup推荐方法，使用户在进行数据服务Mashup时，当用户给出部分的数据服务Mashup后，为其推荐相关的、用户可能感兴趣的数据服务。该方法的目的是减少他们可能进行数据服务Mashup的时间，同时为用户推荐他们可能感兴趣的数据服务，辅助用户构建更加全面的数据服务Mashup。

3　方法概述

在已有的数据服务Mashup方案中，如果两个数据服务的输出通过一个二元操作起来，本文认为这两个数据服务之间存在直接的连接关系。如果多个数据服务同时出现在一定数量的数据服务Mashup方案中，并且对于每个数据服务都存在其他数据服务与其有直接的连接关系，本文认为这多个数据服务之间存在一定的关联关系，属于同一个频繁子集。本文首先对已有的数据服务Mashup方案进行预处理；然后进行图的构造及编码，并利用频繁子图挖掘的方法找到数据服务的频繁子集，形成一个储存数据服务关联关系的知识库；最后在知识库的基础上提出具体的推荐算法。

3.1　划分数据操作单元

由于在数据服务Mashup方案中，用户通常会使用一定的一元操作处理数据服务的输出数据，然后再对其进行组合。如数据服务getOrderInfo与数据服务getReturnInfo存在一个聚合关系，但是与聚合操作join直接联系的是两个filter操作。本文将数据服务及其后续的一元操作看作是一个整体，称之为数据操作单元(data operation unit，DOU)，用数据操作单元对应其所包含的数据服务。

为了找到数据服务之间的连接关系，本文对数据服务Mashup方案进行数据操作单元的划分，用数据操作单元来对应其所包含的数据服务。划分的步骤是：首先遍历整个数据服务Mashup结构找到所有的数据服务，然后以每一个数据服务为源寻找其后续的一元操作直到二元操作或其它数据服务为止，提取出的每个数据服务及其后续的一元操作序列就是该数据服务所对应的数据操作单元。图2a中的虚线部分即为上文的数据服务Mashup方案实例划分的数据操作单元。

从图2a中可以发现，由于用户在对数据服务使用二元操作之后还使用了一元操作进行处理，造成在划分数据操作单元后忽略了一元操作的情况。某些一元操作的丢失会影响数据服务Mashup的输出结果，破坏数据服务Mashup方案的完整性。

为了使划分数据操作单元不影响数据服务Mashup方案本身的完整性，本文利用分配率对数据服务Mashup方案的结构进行等价代换，将聚合操作之后的一元操作提到聚合操作之前、数据服务之后。形式化表示为：

$$\mathrm{UOP}_{C<A>}(\mathrm{AOP}(D_1,D_2))=$$

$$\mathrm{AOP}(\mathrm{UOP}_{C<A1>}(D_1),\mathrm{UOP}_{C<A2>}(D_2)) \tag{1}$$

其中：

$$A\in D_1\cdot\mathrm{Out}\bigcup D_2\cdot\mathrm{Out}$$

$$A1=A\bigcap D_1\cdot\mathrm{Out}\ A_2=A\bigcap D_2\cdot\mathrm{Out}$$

需要进行操作处理的数据实体用D表示，UOP代表一元操作，AOP代表聚合操作。一元操作的具体条件由$C_{<A>}$表示，$A\in D$的属性集合。例如，$d_{\mathrm{C<a1,a2>}}$表示删除数据实体D的a_1,a_2属性。具体证明可以参见文献[12]。

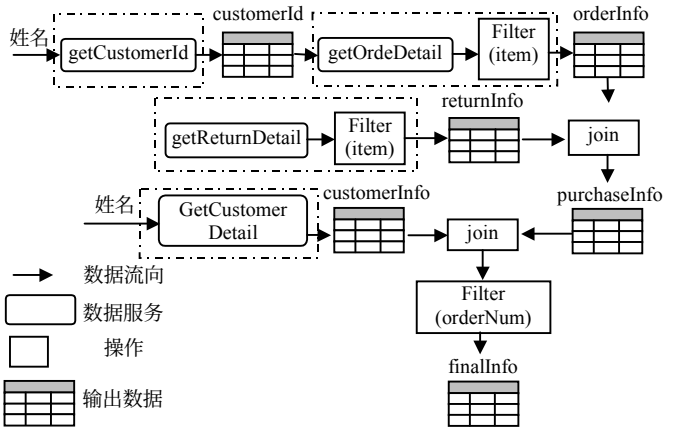
数据服务Mashup方案的结构经过等价变换后，对其进行数据操作单元的划分就能够得到更完整的数据操作单元。图2b是等价变换后的数据服务Mashup方案进行数据操作单元划分，图中filter操作移动到join操作之前，被划分到数据操作单元之中。

3.2　频繁子图挖掘

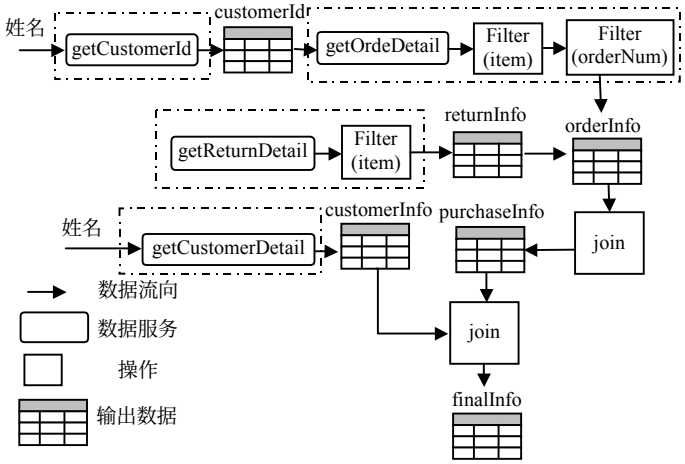
现在有很多频繁子图挖掘的算法，其中基于Apriori性质的包括AGM，FSG，PATH等算法，基于模式增长的主要包括gSpan算法和FFSM算法。文献[16]提出的算法gSpan(graph-based substructure pattern mining)是典型的DFS策略算法，该算法通过对上一代子图扩展频繁边得到候选频繁子图，不仅可以针对无向图，还可以针对有向图进行频繁子图的挖掘[17]。

本文在gSpan算法的基础上进行适应性改进以用于本文中混合图的挖掘。首先将已有的数据服务

Mashup构造成混合图并进行编码。数据服务Mashup所构造的图可以由M=<id, V, E>表示，其中V是M中包含的数据操作单元集合，E是M中包含的所有的连接集合。图3为上文提到的实例所构造的图，其中圆圈代表一个数据操作单元，连线代表数据操作单元之间的连接。



a. 数据服务Mashup等价代换前的数据操作单元划分



b. 数据服务Mashup等价代换后的数据操作单元划分

图2 数据服务Mashup实例中的数据操作单元划分

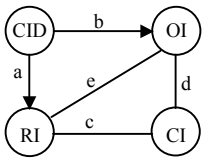


图3 数据服务Mashup构造的图

用户在对相同的数据服务进行处理时，会使用不同的一元操作以及不同的处理顺序，导致同一数据服务对应大量的数据操作单元。为了更好地统计数据操作单元，本文将DS相同的数据操作单元的操作集合取并集得到一个唯一的数据操作单元进行编码。数据操作单元可以由V=<id, DS, Ops, In, Out>表示，其中Ops为DS相同的数据操作单元的操作集合的并集。如对应数据服务getOrderInfo存在包含操作$filter_{<order>}$、$sum_{<order>}$的数据操作单元及包含操作$filter_{<order>}$、$sort_{<order>}$的数据操作单元，这两个数据操作单元可以用<id, getOrderInfo, {$filter_{<order>}$、$sum_{<order>}$, $sort_{<order>}$}, In, Out >来唯一表示。

数据服务Mashup中的连接可以用E=<id, V_{src}id, V_{tar}id, IsD>表示，其中V_{src}id, V_{tar}id用数据操作单元的唯一编码id来表示，IsD表示该连接是有向边还是无向边。数据服务Mashup中的连接用有向边来表示某一数据操作单元的输出作为另一操作数据单元的输入情况，用无向边表示用二元操作将两个数据操作单元的输出组合起来的情况。例如图3中CID的输出可以作为RI的输入，它们之间的连接为有向边<a, CID, RI, true>，而RI与CI是用二元操作组合起来的，它们之间的连接为无向边<a, RI, CI, false>。

对已有的数据服务Mashup进行图的构造以及编码之后，将所有的数据操作单元连接的频度进行排序，得到包含所有数据操作单元以及其频度的集合TV = {<V, freq>}以及包含所有连接以及其频度的集合TE={<E, freq>}。计算连接频度时，判断连接E

和E'是否相同的前提是E和E'均为有向边或均为无向边。如若E和E'均为有向边，要使$E=E'$，必须满足(V_{src}id=V_{src}id′)∩(V_{tar}id= V_{tar}id′)，如果E和E'均为无向边，则满足((V_{src}id=V_{src}id′)∩(V_{tar}id= V_{tar}id′))∪ ((V_{src}id= V_{tar}id′)∩(V_{tar}id= V_{src}id′))。具体的挖掘算法如下：

算法1：频繁子图挖掘

输入：数据组合集合M={<id,V,E>}，连接集合TE，最小支持度minSup，频繁子集最大连接数n

输出：频繁子集集合 S={<s>}

1) TE′ ← getFrequentSet(TE,minSup);
2) orderByLexi(TE′);
3) for each e in TE′
4) S,s ← e;
5) end for
6) While(n>0)
7) n--;
8) s ← subGraph_Extend(S, M)；
9) end while
10) return S

在该算法中，首先将连接中的不频繁的连接去掉获得所有的频繁连接TE′(1))，进行字母排序(2))，初始化为频繁一边图(3)～(5))，将所有的频繁一边图根据包含它的数据服务Mashup按照深度优先遍历的原则进行加边扩展(8))，最终得到连接数不大于n的所有频繁子图以及其所包含的边的频度。频繁子图中所包含的频繁边的频度将作为对用户进行推荐的权重之一。

3.3　具体推荐方法

通过上文步骤可以形成储存数据服务直接连接以及频繁子集的知识库。本文的推荐方法不仅考虑频繁子集，也考虑非频繁的数据服务连接。

首先寻找包含当前数据操作单元的所有频繁子集，按照频繁子集中包含已有数据服务的连接频度相加所得到的数值进行排序，如果频繁子集的数目满足推荐需求，则对其进行截取得到最终的输出列表，否则需要分析非频繁连接，从中获得所有可以与当前的数据服务连接的候选数据服务，根据连接的频率以及可以同当前数据服务连接的数目进行排序，对其进行截取得到最终的推荐列表。

在推荐时优先推荐频繁子集，然后推荐可与当前数据服务进行连接的数据服务，如果频繁子集产生的推荐列表已经满足需求，即推荐列表的大小不小于设置的推荐数目，则不需要再利用非频繁连接进行推荐。

4　实验与效果

4.1　数据集

本文的数据集来源于Yahoo! Pipes及模拟数据集，根据Yahoo! Pipes的数据判断本文提出的推荐方法的准确性，利用模拟数据集检测推荐算法的效率。

Yahoo! Pipes是个服务编辑平台，在该平台上用户创建的服务Mashup称为pipe。在Yahoo! Pipes中定义的Sources以及Operation与本文推荐方法中针对的数据服务操作以及操作等价。所以本文选择Yahoo! Pipes平台上的数据作为本文的实验数据集之一。

本文参照ProgrammableWeb中的APIs和Mashups生成模拟数据集。将数据服务按照关键的输入输出参数分为不同的类别，即由关键输入输出参数来确定类别，关键输入输出参数均不大于3个，为了使不同类别中的数据服务能够关联起来，对于每个类别，至少存在一个其他类别的关键输入与其关键输出存在交集；每个类别中包含的数据服务为10～50个，每个数据服务都包含关键输入输出参数，通过个性的输入输出或不同的数据源进行区别；数据服务与Mashup的分布规则依照二八原则，即80%的Mashup会使用20%的流行数据服务。最终生成的模拟数据集包含20个种类，649个数据服务以及2 000个Mashup。

4.2　实验步骤

本文首先利用Yahoo! Pipes的数据集进行实验检验推荐方法的准确性。随机选取1 200个pipe作为训练集，300个pipe作为测试集。

1) 测试在不同训练集情况下的推荐准确率，训练集的数目为300、600、900、1 200。将测试集分为3组，计算3组的平均得分。

2) 测试在给出不同数目的数据服务的情况下的推荐准确率，在1 200个训练集下，将当前给出的数据服务的数目设为1、2、3、4，测试3组测试集的推荐效果。

利用模拟数据集测试推荐算法的推荐效率。

1) 测试在不同训练集情况下离线生成频繁子集的执行时间，训练集数目设置为300、600、900、1 200、1 500个。

2) 测试在不同训练集以及给出不同数目的数据服务情况下，给出推荐列表所需要的执行时间，测试训练集为600和1 200以及给出的数据服务数目不同下的平均推荐时间。

4.3　实验结果分析

每次推荐数据服务的得分规则如下：推荐算法

给出的候选列表的大小为k，测试集中需要进行推荐总数为n，其中对第i次推荐的数据服务在推荐列表中的排名为$R_i(0 \le R_i \le k$，如果不在推荐列表中则R_i=0)。利用score(i)表示第i个数据服务Mashup中推荐目的数据服务得分，得分公式为：

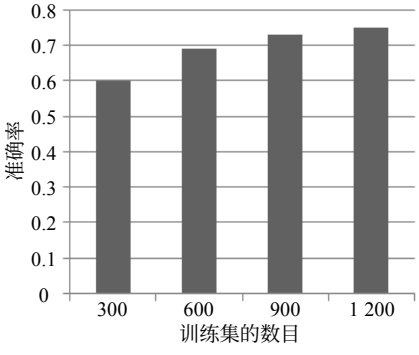
$$\text{score}(i)=\begin{cases} k-R_i & (0<R_i<k) \\ 0 & (R_i=0) \end{cases} \tag{2}$$

推荐过程中会出现推荐目的数据服务集合为空的情况。在计算推荐精度时除去这种情况，因此，实际有效的推荐次数为c<=n。以τ表示推荐算法的精度，则τ的计算公式如下：

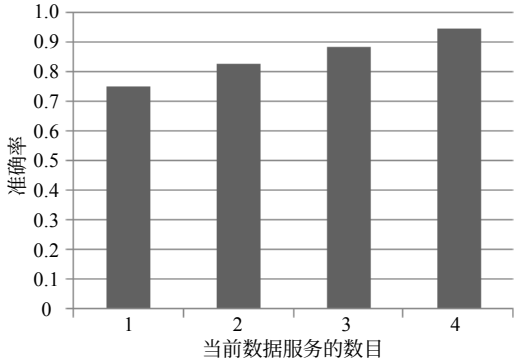
$$\tau=\sum_{i-1}^{n}\text{score}(i)\Big/kn \tag{3}$$

经过多次实验得知，k值设定越大推荐精确度越高，但k值设定过大会导致推荐结果不利于用户选择，故本文试验中将k值设定为7。最终的推荐结果如图4所示。

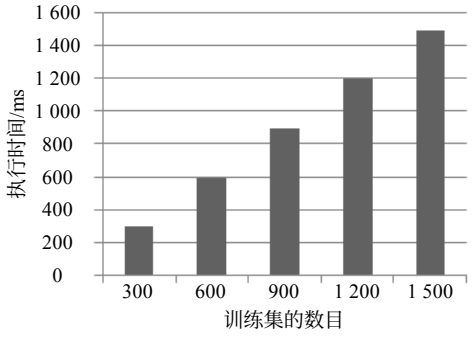
图4a表明在给定数据服务后，随着训练集的增加，推荐精确度逐步升高；图4b表明在训练集大小固定的情况下，用户给出的数据服务数量越多，推荐的精确度越高，表明根据本文的推荐方法能够有效地推荐相关数据服务；图4c和图4d是在模拟数据集下的实验结果，图4c表明随着训练集数目的增加生成频繁子集所需要的时间基本呈现线性增加的趋势，这是由于本文中每次生成频繁子集都需要重新分析所有的Mashup，后续工作中需要考虑避免重复的分析工作；图4d看出训练集为1 200的推荐时间比训练集为600的推荐时间略短，这是由于训练集越多，仅利用频繁子集生成推荐列表的可能性就增加，不需要额外的分析非频繁的连接关系；随着给出的数据服务数量增加，为了全面给出相关数据服务，需要考虑的情况也增加，执行时间相应增加，但还是在可以接受的时间范围内。实验结果表明本文的推荐方法能够在短时间内推荐数据服务。



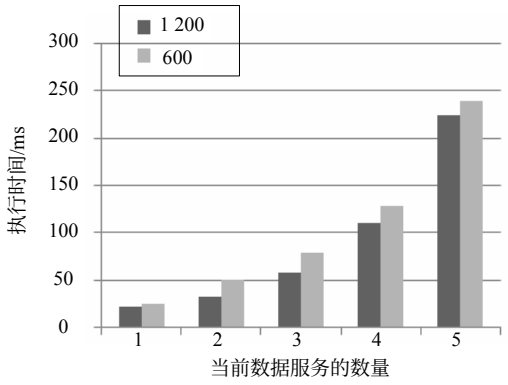
a. 训练集数目不同



b. 给出数据服务数目不同



c. 训练集数目不同



d. 给出的数据服务数目不同

图4 针对两个数据集的推荐效果

5 总 结

本文将数据服务及其后续一元操作处理看作是数据操作单元来代表其所包含的数据服务；以数据操作单元为节点将数据服务Mashup构造成图并编码，统计数据操作单元直接连接的频度并采用gSpan算法挖掘频繁子图找到数据操作单元的频繁子集；通过数据操作单元之间的关联关系来对应数据服务之间的关联关系并生成知识库；在通过历史记录所构造的知识库基础上提出一个推荐算法，在用户给出部分数据服务Mashup后，推荐给用户相关的后续数据服务列表；最后通过实验验证了推荐方法的有效性。将来的研究重点将放在改善推荐方法性能、

研究数据服务之间的依赖性来找到潜在的关联关系、结合数据流的服务生成、优化及推荐上。

参 考 文 献

[1] HOANG D D, PAIK H Y, BENATALLAH B. An analysis of spreadsheet-based services mashup[C]//Australasian Conference on Database Technologies. Australia: Australian Computer Society, 2010: 141-150.

[2] ALTINEL M, BROWN P, CLINE S, et al. Damia: a data mashup fabric for intranet applications[C]//the 33rd International Conference on Very Large Data Bases. Austria: VLDB Endowment, 2007:1370-1373.

[3] HAN Yan-bo, WANG Gui-ling, JI Guang, et al. Situational data integration with data services and nested table[J]. Service Oriented Computing and Applications, 2013, 7(2): 129-50.

[4] JHINGRAN A. Enterprise information mashups: integrating information, simply[C]//the 32nd International Conference on Very Large Data Bases. Korea: VLDB Endowment, 2006: 3-4.

[5] LIU Xuan-zhe, GANG Huang, HONG Mei. Discovering homogeneous web service community in the user-centric web environment[J]. IEEE Transactions on Services Computing, 2009, 2(2): 167-181.

[6] WANG Gui-ling, YANG Shao-hua, HAN Yan-bo. Mashroom: End-user mashup programming using nested tables[C]//International World Wide Web Conference Committee (IW3C2). Madrid, Spain: ACM, 2009: 861-870.

[7] HAN Yan-bo, WANG Gui-ling, JI Guang, et al. Situational data integration with data services and nested table[J]. Service Oriented Computing and Applications, 2013, 7(2): 129-150.

[8] Yahoo. Yahoo pipes[EB/OL]. [2014-12-19]. http://pipes.yahoo.com/pipes/.

[9] ALTINEL M, BROWN P, CLINE S, et al. Damia: a data mashup fabric for intranet applications[C]//Proceedings of the 33rd International Conference on Very Large Data Bases. Austria: VLDB Endowment, 2007: 1370-1373.

[10] WANG Gui-ling, ZHANG Sai, LIU Cheng, et al. A dataflow- pattern-based recommendation approach for data service mashups[C]//Services Computing(SCC), 2014 IEEE International Conference. USA: IEEE, 2014.

[11] Programmable Web. Browse the world's largest API repository[EB/OL].[2014-12-19]. http://www.programmableweb.com/category/all/apis.

[12] GREENSHPAN O, MILO T, POLYZOTIS N. Autocompletion for mashups[J]. Proceedings of the VLDB Endowment, 2009, 2(1): 538-49.

[13] MA Y, LU X, LIU X, et al. Data-driven synthesis of multiple recommendation patterns to create situational Web mashups[J]. Science China Information Sciences, 2013, 56(8): 1-16.

[14] YANG Jian-yu, HAN Jun, WANG Xu, et al. MashStudio: an on-the-fly environment for rapid mashup development [J]. Lecture Notes in Computer Science, 2012, 7646: 160-173.

[15] LIN Hai-lun, ZHANG Cheng, ZHANG Peng. Web technologies and applications[M]. Berlin Heidelberg: Web Technologies and Applications. Springer. 2012.

[16] YAN Xi-feng, HAN Jia-wei. GSPAN: Graph-based substructure pattern mining; proceedings of the data mining[C]//IEEE International Conference on Data Mining. Maebashi City. Japan: IEEE Press, 2002: 721-724.

[17] 周溜溜，业宁. 基于对gSpan改进的有向频繁子图挖掘算法[J]. 南京大学学报: 自然科学版, 2011, 47(5): 532-543.
ZHOU Liu-liu, YE Ning. Digraph frequent subgraph mining based on gSpan[J]. Journal of Nanjing University: Natural Sciences, 2011, 47(5): 532-543.

编 辑 税 红