面向LBS的服务匹配方法研究

张德干 王 冬
(天津理工大学智能计算及软件新技术天津市重点实验室 天津 300384)
(天津理工大学计算机视觉与系统教育部重点实验室 天津 300384)

摘 要 随着普适计算的不断深入发展,基于位置的服务也逐渐成为应用研究的重要方面。服务发现技术成为核心问题,而服务匹配就是重中之重,也是现在研究的热点问题。在对用户提供服务时,不仅要提高服务质量,而且要充分考虑用户的偏好。在面向位置服务领域分析与比较了现有的主流服务发现协议,并基于分析比较,提出了一种新的服务匹配方法,它通过分层匹配,以及充分考虑用户喜好程度来提高服务匹配成功率,从而达到提高服务质量的目的。

关键词 普适计算,服务发现,服务匹配
中图法分类号 TP311 **文献标识码** A

Research on Service Matching Method for LBS

ZHANG De-gan WANG Dong
(Tianjin Key Lab of Intelligent Computing & Novel Software Technology, Tianjin University of Technology, Tianjin 300384, China)
(Key Laboratory of Computer Vision and System of Ministry of Education, Tianjin University of Technology, Tianjin 300384, China)

Abstract With the development of pervasive computing, location-based service is becoming an important aspect of applied research, and the technology of service discovery becomes a core issue. So service matching is the most important in service discovery, also it became hot spot in researching. In the process of providing service to user, not only the quality of service should be improved, but also user's preferences should be considered. The existing mainstream service discovery protocols in the location-oriented service were compared. Based on analysis and comparison, a new algorithm of service matching was proposed. It improves the service matching success rate through hierarchical matching and full accounting of the user's preference, and ultimately aims to improve service quality.

Keywords Pervasive computing, Service discovery, Service matching

1 引言

普适计算旨在建立一个充满感知、计算和通信能力的环境,并使这个环境与人们的生活逐渐融合在一起[1]。情境感知是达到这一目标的关键,其任务是使普适计算设备在用户需要时能够提供适合于当时任务、地点和时间的服务[2]。当用户的状态因时、因地、因环境的不同而不断变化时,这些服务表现出显著的随时特性,最终体现在服务的及时性上。"情境感知计算"(context-aware computing)的重要分支——位置感知计算(location-aware computing)是近年来国内外学者研究的热点。随着移动通信技术的迅速发展、手持无线终端性能的不断提升、3G网络不断推广以及4G标准的提出,无线应用领域正面临着前所未有的新机遇。

基于位置的服务的出现和发展与移动技术的进步和市场的需求有着密切的关系[3]。LBS,即基于位置的服务(Location Based Service, LBS)是一种在基于当前或者已知位置上提供信息服务的平台。它利用GIS技术、空间定位技术和网络通信技术,为移动对象提供基于空间地理位置的信息服务。LBS技术的核心目标就是使用户可以在任何时间、任何地点获得基于定位信息的地理信息服务。它需要融合信息技术中的诸多新技术,把位置作为相关信息的索引,为用户提供与位置相关的信息服务。LBS服务几乎涵盖了人类动态活动的每一方面:安全、防卫、紧急事故、导航、生活便利、娱乐、旅行助理、后勤、移动资产管理等。目前主要通过移动通信网络获取移动终端用户的位置信息,在电子地图平台的支持下,为用户提供相应服务的一种增值业务。

目前,已经有许多服务发现方面的协议[4],诸如Jini、UPnP、Salutation、SLP等[5-10]。

Jini[11,12]是SUN在Java继承上建立的面向对象的服务发现机制,主要实体为服务、客户和查找服务。在Jini中,服务发现通过接口匹配和属性匹配进行,它根据服务的类型或属性向查找服务查询合适的服务,然后查找服务把查询的结果返回给客户。UPnP[13]全名是Universal Plug and Play,主要是微软推行的一个标准,定义了SSDP(Simple Service Dis-

到稿日期:2011-04-15 返修日期:2011-09-28 本文受国家自然科学基金项目(60773073,61001174,61170173),教育部新世纪优秀人才计划项目(NCET-09-0895),天津市自然科学基金项目(10JCYBJC00500)资助。
张德干(1969-),男,博士,教授,主要研究方向为移动计算、智能控制、网络通信,E-mail:gandegande@126.com;王 冬(1986-),女,硕士生,主要研究方向为移动计算、位置服务。

covery Protocol)来处理服务发现问题。它定义了服务和设备的类型、属性和行为模板,模板文件的定义以及设备和服务的描述都基于 XML 语言。服务的访问遵循 SOAP(Simple Object Access Protocol)协议。Salutation[14] 是 Salutation Consortium 发布的一项技术,它集服务发现和会话管理于一身,是一种与操作系统、通讯协议和硬件平台都无关的开放标准。SLP[15](Service Location Protocol)是 IETF 的 SRVLOC 工作组于 1997 年制定的标准,是基于 TCP/IP 协议簇建立的服务发现模式。

而在 LBS 中,如何进行服务的发现,如何更加有效地提供服务,也是关键问题。本文正是针对 LBS 中的服务发现的关键技术,即服务匹配进行研究和应用。

2 现有服务发现协议存在的问题

上面已经提到了目前已有的服务发现协议,现有的几种服务发现机制在设计上都有各自的优缺点,每一种都有其适用的范围,但是也有以下几点不足:

(1)没有增加语义信息

在已有的服务发现协议中,服务的描述通过服务的名称或有限的属性集合来完成,缺乏必要的语义信息。服务匹配也仅是基于接口或属性集来进行,不能推理服务的功能和特性,服务与服务之间不能互相识别,并且容易出现一词多义和同词异义的情况。

(2)不支持上下文相关

目前的服务发现协议都不支持与上下文相关的应用,其服务时捕获的只是静态属性。

(3)不适合公用广域网

已有的服务发现协议很多依赖于 IP 组播。

(4)安全及隐私性不足

没有考虑到公共环境下的服务发现的安全性和隐私性等要求。

(5)不太适合于移动网络

现有的服务发现协议中很多都需要有一个核心的服务器来维持服务目录,而这在无线网络中代价花费大,也不符合普适计算环境。

综上所述,现有的服务发现协议虽然对于一般意义上的服务发现问题提供了很好的解决方法,但还缺乏对更为灵活的服务发现的支持,同时缺乏更为有效的安全机制。

另外,如应用到 LBS 中,是根据位置和用户的喜好程度来进行服务的,在此还要考虑到用户的偏好问题,这样能更加体现以人为中心的思想。因此,本文主要研究如何提出一种更为有效的服务匹配算法来改进服务发现。

3 服务匹配算法

目前已经有很多服务匹配方面的算法研究,其中比较有影响力的是 Paolucci 等人[16] 提出的弹性匹配算法。

3.1 弹性匹配算法

弹性匹配算法[17] 主要是基于功能性的服务匹配,考虑了服务的输入、输出参数的匹配。定义 4 种匹配等级:完全匹配(Exact)、插拔匹配(Plugin)、包含匹配(Subsume)、匹配失败(Fail)。细分之下有 5 种关系[18],如图 1 所示。其中,R 表示服务请求者所需的服务,S 表示服务提供者所提供的服务。

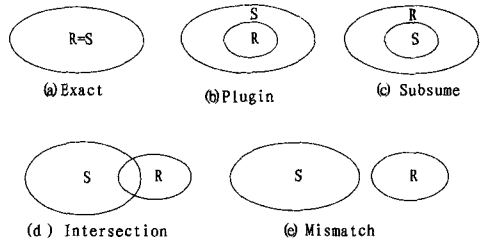

图 1 服务提供者 S 和服务请求者 R 之间的关系图

弹性匹配算法描述如下:

```
degreeOfMatch(outR,outA)
{   if(outA==outR) return exact;
    else if(ourR subclassOf outA) return exact;
    else if(outA subsumes outR) return plugin;
    else if(outR subsumes outA)
      return subsumes;
    else return fail;
}
```

这里表达的是服务输出之间的匹配。其中,outR 表示用户请求的输出,outA 表示服务提供的输出,subclassOf 表示子类关系,subsumes 表示包含关系。

弹性匹配算法虽然可以解决请求服务与发布服务之间的匹配,但是它只考虑了输入与输出之间的匹配,不能更加准确地定位到所需的服务。

3.2 新的服务匹配算法的设计

服务匹配主要有功能性的匹配和非功能性的匹配[19,20]。本文基于弹性匹配算法的思想,定义一个函数 ConceptMatch()来表示上述 4 种匹配层次。它也是以弹性匹配算法为基础的,只是把一个概念是另一个概念的直接子类归为插拔匹配,而不是完全匹配。

其定义如下:

(1)完全匹配(exact) 当请求概念与提供概念相等时;

(2)插拔匹配(plugin) 当提供概念包含请求概念时;

(3)包含匹配(subsume) 当请求概念包含提供概念时;

(4)匹配失败(fail) 不属于以上 3 种情况的为匹配失败。

定义一个服务提供者所提供的服务 AD 为以下一个五元组:

$$AD=\{AD_{type},AD_{input},AD_{output},AD_{QoS},AD_{interest}\}$$

式中,AD_{type} 为服务提供者提供的服务类型,AD_{input} 为该服务的输入参数,AD_{output} 为服务的输出参数,AD_{QoS} 为该服务的 QoS 参数,$AD_{interest}$ 为该服务的用户喜好度。

同样,一个服务请求者所请求的服务 Req 也定义为一个如下五元组:

$$Req=\{Req_{type},Req_{input},Req_{output},Req_{QoS},Req_{interest}\}$$

式中,Req_{type} 为服务请求者所请求的服务类型,Req_{input} 为该服务的输入参数,Req_{output} 为该服务的输出参数,Req_{QoS} 为该服务的 QoS 参数,$Req_{interest}$ 为该服务的用户喜好度。

首先,通过服务类型匹配过程可以把所有不匹配的服务过滤掉,把准确的和相似的匹配服务提供给下一步的匹配,即给服务输入输出匹配。在这个匹配过程中就看服务请求与广告服务是否引用的是同一本体,若不是,可以直接过滤掉候选服务;若是,就直接应用函数:

ConceptMatch(*Req*. type,*AD*. type)

若为 Fail,过滤掉该候选服务,否则可以进行下一步的匹配。

3.2.1 服务输入输出匹配度计算

进行服务输入输出匹配度计算,实际上就是表现在两个本体的概念之间的相似度问题。在一个本体概念树 T 中,其父概念直接包含子概念,设其根节点为 R,C_1 和 C_2 为两个子节点,则定义概念 C_1,C_2 的相似度为:

$$\mathrm{Sim}(C_1,C_2)=\begin{cases}1, & C_1=C_2\\ \dfrac{1}{S_1}\prod_{i=1\cdots n}\dfrac{1}{S_{ai}}, & \text{存在一条路径 } C_1,C_{a1},C_{a2},\cdots,C_{an},C_2,\\ & S_1 \text{ 为 } C_1 \text{ 的子类数},S_{ai} \text{ 为 } C_{ai} \text{ 的子类数}\\ 0, & C_1,C_2 \text{ 之间不存在一条路径且 } C_1\neq C_2\end{cases} \tag{1}$$

如是多个输入输出,计算其平均值即可。

这样可算得输入输出的匹配相似度,记为 IOMatch(*AD*, *Req*),且

$$\mathrm{IOMatch}(AD,Req)=\frac{InSim(AD,Req)+OutSim(AD,Req)}{2} \tag{2}$$

3.2.2 QoS 参数匹配

在服务 QoS 参数匹配中,在此考虑服务成本、响应时间和可靠性 3 个方面,QoS=⟨*cost*,*time*,*reliability*⟩。QoS 的参数可以分为两类:一类是质量度量的值越大,代表该质量参数越优,如可靠性;另一类是质量度量的值越小,代表该质量参数越优,如响应时间、服务花费。为了方便计算和统一,在这 3 个参数中,仅可靠性参数是希望度量值越大越好,所以考虑其倒数$\frac{1}{reliablity}$。如果 *reliability* 为 0 的话,意味着没有可靠性,可以直接筛选掉该服务。这样,将 QoS 转化为 QoS′=⟨*cost*,*time*,$\frac{1}{reliablity}$⟩,三者统一考虑度量值越小越优。可按如下公式计算三者的指标数值:

$$P_{ij}=\begin{cases}\dfrac{Q_j'^{\max}-Q_j'}{Q_j'^{\max}-Q_j'^{\min}}, & \text{若 } Q_j'^{\max}-Q_j'^{\min}\neq 0\\ 1, & \text{若 } Q_j'^{\max}-Q_j'^{\min}=0\end{cases} \tag{3}$$

式中,$Q_j'^{\max}$代表 QoS′中某项质量参数度量的最大值,$Q_j'^{\min}$代表 QoS′中某项质量参数度量的最小值,$1\leqslant i\leqslant n$,$1\leqslant j\leqslant 3$,其中 i 表示第 i 个服务,j 表示第 j 个质量度量。为了比较请求服务的质量参数和发布服务质量参数对应的度量的差异值大小,这里的 $Q_j'^{\max}$、$Q_j'^{\min}$不是单独指发布服务中的最大值和最小值,而是包含请求服务的质量参数在内的最大和最小值。$Q_{i,j}'$指的是各个发布服务中的 QoS 转化为 QoS′对应的服务质量参数度量。

得到对应的指标后,可对每个服务进行规范化,如式(4)所示:

$$S_{\mathrm{QoS}}(s_i)=\sum_{j=1}^{3}(P_{ij}*W_j) \tag{4}$$

式中,W_j 是服务使用者加在各项质量参数上的一个权重,满足 $W_j\in[0,1]$,$\sum W_j=1$。

如服务请求者对所查找服务的 QoS 要求为(100,300,0.8);现有 3 个广告服务 *AD*1、*AD*2、*AD*3,它们的质量参数度量分别为(20,30,0.9)、(45,50,0.8)、(30,60,0.9)。假设 3 个质量参数的权重分别为 0.3,0.3,0.4。

首先可以将其转化为(100,300,10/8)与 3 个广告服务(20,30,10/9)、(45,50,10/8)、(30,60,10/9),并且按照式(3)可得:

对 $AD1$:$P_{11}=1$,$P_{12}=1$,$P_{13}=1$;

对 $AD2$:$P_{11}=11/16$,$P_{12}=25/27$,$P_{13}=0$;

对 $AD3$:$P_{11}=7/8$,$P_{12}=8/9$,$P_{13}=1$。

所以,

QoSMatch(*AD*1,*Req*)

$=S_{\mathrm{QoS}}(s_1)=1*0.3+1*0.3+1*0.4=1$;

QoSMatch(*AD*2,*Req*)

$=S_{\mathrm{QoS}}(s_2)=11/16*0.3+25/27*0.3+0*0.4$

$=0.48403$

QoSMatch(*AD*3,*Req*)

$=S_{\mathrm{QoS}}(s_3)=7/8*0.3+8/9*0.3+1*0.4=0.9292$

若用户设定的服务质量相似度阈值为 0.75,则返回的满足用户需求的服务为 *AD*1、*AD*3。

3.2.3 用户喜好度的匹配

根据上面提到的 QoS 的匹配可知,对于用户喜好度它是个积极指标。这个参数也充分考虑到了在服务中以人为中心,充分考虑到了用户的感受。这个指标当然也是越大越好,所以可以根据式(5)得其匹配度:

$$S_{\mathrm{interest}}=\begin{cases}\dfrac{AD_{\mathrm{interest}}-I^{\min}}{I^{\max}-I^{\min}}, & \text{若 } I^{\max}-I^{\min}\neq 0\\ 1, & \text{若 } I^{\max}-I^{\min}=0\end{cases} \tag{5}$$

式中,$I^{\min}$和 $I^{\max}$代表发布服务和包含请求服务在内的用户喜好度的最小和最大值,AD_{interest}为各个发布服务中的对应的用户喜好度的指标。

3.2.4 服务的综合匹配

综合上面各阶段的匹配,可得到服务的综合匹配。可用式(6)来计算服务的综合匹配:

$$\begin{aligned}\mathrm{ServiceMatch}(Req,AD)=&w_1*\mathrm{IOMatch}(Req,AD)+\\ &w_2*S_{\mathrm{QoS}}(Req,AD)+(1-\\ &w_1-w_2)*S_{\mathrm{interest}}(Req,AD)\end{aligned} \tag{6}$$

式中,w_1,w_2 分别是 IOMatch(*Req*,*AD*)和 $S_{\mathrm{QoS}}(Req,AD)$所占的权重。

4 服务匹配算法的具体描述

在实际匹配中,将请求服务和所有发布的服务进行依次匹配,只有通过上一层匹配的服务才能进入下一层匹配,使服务集合逐步缩小,最终得到满足用户请求的服务或服务集合。图 2 给出了服务匹配的流程图。

本文从 4 个方面进行匹配,输出满足用户指定阈值的服务,并按相似度降序排列。服务匹配算法的伪代码如下:

```
List matchList(AD,Req)
{ for each AD in ADServices do
  {
    if (ConceptMatch(Req. type,AD. type)! =fail)
      {//k1,k2,k3 分别为用户指定的输入输出、QoS、用户喜好度的匹配阈值
        if(IOMath(Req,AD)≥k1
    &&SQoS(Req,AD)≥k2&&Sinterest(Req,AD)≥k3)
              matchList. append(AD);
```

```
        }
    }
    matchListResult=sort(matchList); //按照相似度的大小降序排列
        return matchListResult;
    }
```

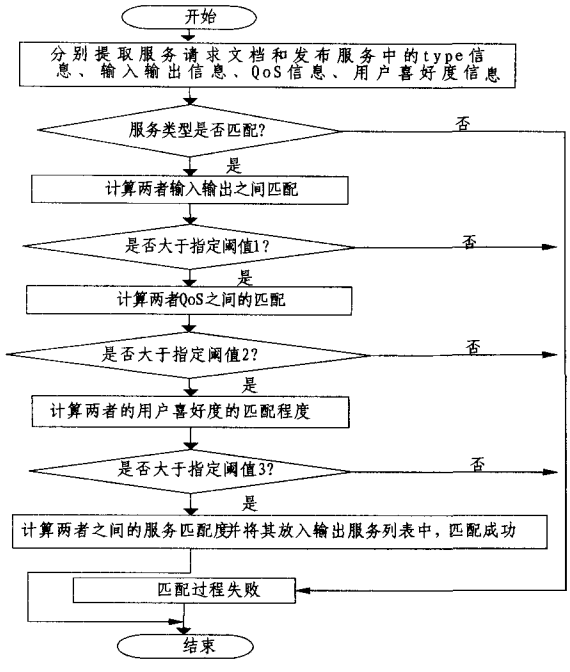

图 2　服务匹配的流程图

服务排序算法 sort( )的伪代码如下：

```
sort(AD1,AD2)
    {
if(ServiceMatch(Req,AD1)>ServiceMatch(Req,AD2)),则输出顺序为 (AD1,AD2);
if(ServiceMatch(Req,AD1)<ServiceMatch(Req,AD2)),则输出顺序为 (AD2,AD1);
if(ServiceMatch(Req,AD1)==ServiceMatch(Req,AD2))
{
if(IOMatch(Req,AD1)>IOMatch(Req,AD2)),则输出顺序为(AD1,AD2);
if(IOMatch(Req,AD1)<IOMatch(Req,AD2)),则输出顺序为(AD2,AD1);
if(IOMatch(Req,AD1)==IOMatch(Req,AD2)),则输出顺序随意;
        }
    }
```

5　实例应用分析

下面分析一个可以应用到 LBS 中的例子。假设 Req 是请求服务，AD1、AD2、AD3 是发布服务的描述，具体表示如下：

```
Req:type:Travel
    input:title
    output:Logging
    QoS information:cost:100 time:300 reliablity:0.8
    interest information:0.8
AD1:type:Travel
    input:title
    output:Hotel information
    QoS information:cost:20 time:30 reliablity:0.9
    interest information:0.9
AD2:type:Entertainment
    input:title
    output:Music information
    QoS information:cost:45 time:50 reliablity:0.8
    inertest information:0.8
AD3:type:Site
    input:title
    output:Travel
    QoS information:cost:30 time:60 reliability:0.9
    interest information:0.9
    QoS information:cost:30 time:60 reliability:0.9
    interest information:0.9
```

该例子引用了如图 3 所示的本体。

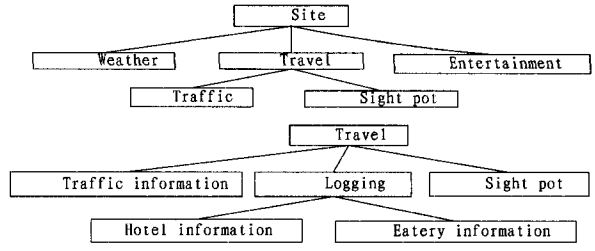

图 3　所引用到的本体示意图

对于 *Req* 和 *AD*1 之间的匹配，首先是服务类型匹配：

ConceptMatch(*Req*. type,*AD*1. type)=exact

可以进行下一步的匹配。在输入输出匹配时，可以看到：

ConceptMatch(*Req*. output,*AD*1. output)=subsume

根据式(1)可得其输出匹配相似度为 0.5，加上两者的输入相同，总的 IOMatch(*Req*,*AD*1)=0.75；若用户设定的输入输出阈值为 0.6，则可以进行下一轮匹配；又 QoSMatch(*Req*,*AD*1)=1。通过式(5)可以计算出喜好程度的匹配度 $S_{interest}=1$，若 $w_1=w_2=0.3$，则 ServiceMatch(*Req*,*AD*1)=0.75 * 0.3+0.3 * 1+0.4 * 1=0.925。

对于 *Req* 和 *AD*2，它们在类型这一项就不一样，可以直接过滤掉。

对于 *Req* 和 *AD*3 之间的匹配，首先是服务类型匹配：

ConceptMatch(*Req*. type,*AD*3. type)=plugin

可以进行下一步的匹配。在输入输出匹配时，可以看到：

ConceptMatch(*Req*. output,*AD*3. output)=plugin

根据式(1)可得其输出匹配相似度为 1/6，加上两者的输入相同，总的 IOMatch(*Req*,*AD*3)=0.583；若用户设置的输入输出匹配度为 0.6，则低于这个权值，没法进入下一轮的匹配，筛选掉该服务。

那么最后返回给用户的服务列表为 *AD*1。

由上述分析知，本匹配算法将每个匹配的结果都用一个介于[0,1]之间的数值表示，只要达到用户指定的阈值，就可以输出给用户；并且，本算法不是单从服务类型进行过滤，除了考虑服务的输入输出参数的匹配，还考虑了 QoS 的匹配，更重要的是考虑了用户的喜好度，这就使得查找到的服务不仅能满足用户的请求，而且接近用户的喜好，从而更加满足用户的需求。综合以上几个方面，本算法整体提高了服务发现的查准率。

(下转第 61 页)

Press,2008:11-20
[12] Jøsang A. A logic for uncertain probabilities[J]. International Journal of Uncertainty, Fuzziness and Knowledge-based Systems,2001,9(3)
[13] Huang Jing-wei, Nicol D. A Formal-Semanti- cs-Based Calculus of Trust[J]. Internet Computing, IEEE,2010,14(5):38-46
[14] Loscocco P, Smalley S. Integrating flexible support for security policies into the Linux operating system[R]. U. S. National Security Agency(NSA). Feb. 2001
[15] Wang H M, Tang Y B, Yin G, et al. Trustworthiness of Internet-based software[J]. Science in China Series F: Information Sciences,2006,49(6):759-773
[16] TCG Specication Architecture Overview[S]. TCG, April 2004: 11-12
[17] Nguyen E A T, Weiss J, Watson J, et al. Toward an approach to measuring software trust[C]// IEEE Symposium on Security and Privacy. 1991:198-218
[18] Avizienis A, Laprie J C, Randell B, et al. Basic concepts and taxonomy of dependable and secure computing[J]. IEEE Trans. on Dependable and Secure Computing,2004,1(1):11-33
[19] Mundie C, de Vries P, Haynes P, et al. Trustworthy computing [R]. Microsoft White Paper. 2002. http://download. microsoft. com/download/a/f/2/af22fd56-7f19-47aa-8167-4b1d73c d3c57/twc_mundie. doc
[20] 李小勇,桂小林,等. 基于行为监控的自适应动态信任度测模型[J]. 计算机学报,2009(4):664-674

(上接第 22 页)

同基于关键字的服务发现相比,引入语义本体后,本体概念间的等价和继承关系可以使相关的实体在输入输出匹配时进行互相匹配。另外可以设定不同的匹配阈值,以用户的选择来缩放服务选择的范围。由此看来,本算法增加了可能匹配服务的范围,使得服务的查全率也有所提高。

下面是通过构造的 100 组实验数据而得出的查全率和查准率的对比图。其中,图 4 是本文算法和弹性匹配算法的查准率对比图,图 5 是本文算法和弹性匹配算法的查全率对比图。

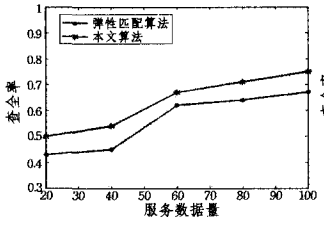

图 4 查准率对比图

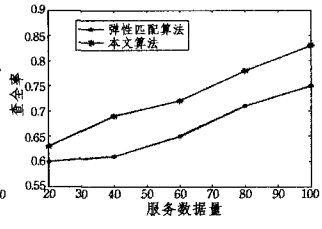

图 5 查全率对比图

结束语 通过分析现有的服务发现方法,找出在具体应用到普适计算中的位置服务方面的不足之处,然后提出一种新的服务匹配算法。本算法从性能上提高了查准率,并且考虑了用户的喜好度和服务质量,更加人性化。但是在服务发现的安全性和隐私性以及完善匹配算法上,没有做进一步的研究,加上构建一个具体的 LBS 系统,也是接下来工作的重要方面。

参 考 文 献

[1] Weiser M. The computer for the twenty-first century[J]. Scientific American,1991,265(3):94-104
[2] 徐光祐,史元春,谢伟凯. 普适计算[J]. 计算机学报,2003,26(9):1042-1050
[3] Kaasinen E. User needs for location-aware mobile services[J]. Per Ubiquit Comput,2003(7):70-79
[4] 周晓,沈振宇,陈鸣. 服务发现机制的比较与分析[J]. 计算机工程与科学,2003,25(2):56-60
[5] Yoon Y-B, Youn H-Y. A New Service Discovery Scheme Adapting to User Behavior for Ubiquitous Computing[M]. Berlin Heidelberg: Springer-Verlag,2005:19-28
[6] Kim M C, Kim S J. A Scenario-based User-oriented Integrated Architecture for Supporting Interoperability Among Heterogeneous Home Network Middlewares[M]. Berlin Heidelberg: Springer-Verlag,2006:669-678
[7] d'Amorim M, Ferrazn C. A Design for Jtrader, an Internet Trading Service[M]. Berlin Heidelberg: Springer-Verlag,2001: 159-166
[8] Oprescu J, Rousseau F, Duda A. Push Driven Service Composition in Personal Communication Environments[Z]. International Federation for Information Processing(IFIP). 2003:505-510
[9] Lee S-H, Jang K-S, Shin D-R. Agent-based Discovery Middleware Supporting Interoperability in Ubiquitous Environments [M]. Berlin Heidelberg: Springer-Verlag,2007:141-149
[10] Hasselmeyer P. On Service Discovery Process Types[M]. Berlin Heidelberg: Springer-Verlag,2005:144-156
[11] Arnold K, O'Sullivan B, Scheifler R W, et al. The Jini specification[S]. Addison-Wesley,1999
[12] The Jini device architecture specification[S]. Sun Microsystem, 1999
[13] Universal Plug and Play (UpnP) [EB/OL]. http://www. upnp. org
[14] Miller B A, Pascoe R A. Salutation Service Discovery in Pervasive Computing Environments[R]. IBM Pervasive Computing White Paper. February 2000
[15] Guttman E, Perkins C E, Veizades J, et al. Sevice Location Protocol[R]. Version 2, IETF, RFC 2680. June 1999
[16] Paolucci M, Kawamura T, Payne T R, et al. Semantic Matching of Service Capabilities[C]//Proceedings of 1st International Semantic Web Conference (ISWC2002). Berlin: Springer-Verlag, 2002:333-34
[17] 林清滢,彭文灵. 基于 OWL-S 的 Web 服务匹配方法及其实现[J]. 微计算机应用,2006,27(4):496-498
[18] Li Lei, Horrocks I. A software framework for matchmaking based on semantic Web Technology[C]// Proceedings of the Twelfth Inernational World Wide Web Conference (WWW 2003). Budapest: ACM Press,2003:31-339
[19] 吕庆聪,周集良,杨帆,等. 普适计算服务匹配技术研究[J]. 计算机科学,2009,36(11):182-185
[20] Mokhtar S B. EASY: Efficient semantic service discovery in pervasive computing environments with QoS and context support [J]. J Syst Soft-ware,2007,10:9-11