

SHANGHAIUNIVERSITY

毕业设计（论文）

**UNDERGRADUATEPROJECT(THESIS)**

**题目:****基于回归分析的交通拥堵预警系统**

|  |  |
| --- | --- |
| **学院** | **计算机工程与科学学院** |
| **专业** | **计算机科学与技术** |
| **学号** | **15121252** |
| **学生姓名** | **卢广犇** |
| **指导教师** | **吴绍春** |
| **起讫日期** | **2019.02.25 – 2019.06.07** |

目录

[摘要 III](#_Toc460951271)

[ABSTRACT IV](#_Toc460951272)

[第1章绪论 1](#_Toc460951273)

[§1.1微博用户模型构建的背景及意义 1](#_Toc460951274)

[§1.2用户模型研究现状及存在的问题 1](#_Toc460951275)

[§1.2.1研究现状 1](#_Toc460951276)

[§1.2.2微博存在的问题 2](#_Toc460951277)

[§1.2.3微博用户模型构建的难点 2](#_Toc460951278)

[§1.3本文研究内容及目标 2](#_Toc460951279)

[§1.3.1研究内容 2](#_Toc460951280)

[§1.3.2研究目标 3](#_Toc460951281)

[§1.4本文组织结构 3](#_Toc460951282)

[第2章基于维基百科的本体构建方法 4](#_Toc460951283)

[§2.1本体概述 4](#_Toc460951284)

[§2.1.1本体的基本概念 4](#_Toc460951285)

[§2.1.2本体构建方法 5](#_Toc460951286)

[§2.2基于维基百科的本体构建 6](#_Toc460951287)

[§2.2.1维基百科简介 6](#_Toc460951288)

[§2.2.2传统方式构建本体存在的问题 7](#_Toc460951289)

[§2.2.3基于维基百科的本体构建 7](#_Toc460951290)

[§2.3本章小结 9](#_Toc460951291)

[第3章基于本体的微博用户模型构建方法 10](#_Toc460951292)

[§3.1用户模型概述 10](#_Toc460951293)

[§3.2用户模型表示方法 10](#_Toc460951294)

[§3.2.1常用用户模型表示方法 11](#_Toc460951295)

[§3.2.2本体用户模型表示方法 11](#_Toc460951296)

[§3.3微博用户模型的构建方法 12](#_Toc460951297)

[§3.3.1特征词提取 12](#_Toc460951298)

[§3.3.2兴趣度计算 14](#_Toc460951299)

[§3.3.3用户模型的生成 14](#_Toc460951300)

[§3.4本章小结 16](#_Toc460951301)

[第4章微博用户模型构建系统设计与实现 17](#_Toc460951302)

[§4.1系统整体设计 17](#_Toc460951303)

[§4.1.1开发环境简介 17](#_Toc460951304)

[§4.1.2系统整体架构 17](#_Toc460951305)

[§4.2数据库设计 18](#_Toc460951306)

[§4.3功能模块设计 22](#_Toc460951307)

[§4.3.1本体的构建与显示 22](#_Toc460951308)

[§4.3.2用户基本数据显示 23](#_Toc460951309)

[§4.3.3用户模型构建与显示 24](#_Toc460951310)

[§4.4本章小结 25](#_Toc460951311)

[第5章总结与展望 26](#_Toc460951312)

[§5.1本文总结 26](#_Toc460951313)

[§5.1.1本文的主要工作 26](#_Toc460951314)

[§5.1.2本文的主要创新点 26](#_Toc460951315)

[§5.2展望 26](#_Toc460951316)

[致谢 28](#_Toc460951317)

[参考文献 29](#_Toc460951318)

[附录：部分源程序清单 31](#_Toc460951319)

基于回归分析的交通拥堵预警系统

# 摘要

XXX。

关键词：XXXX，XXX，XXX，XXX

Traffic congestion warning system based on regression analysis

# ABSTRACT

In recent years, with the rapid development of Micro-blog, the need that users gain the access to information is also a linear growth momentum. The amount of Sina Micro-blog registered users has reached 503 million by 2012. However, the daily flood of Micro-blogs has a serious impact on the quality of information users receive. Thus, how to find content that they are interested in quickly and accurately? Or can we push the information according to the user's interest actively? That is what this paper concerns.

User model is a formal description of users' interests. To establish a precise user model for Micro-blog users, in order to recommend the information they concerned about and contents they are interested in, is the development trend of personalized recommendation. The achievements of this paper can be used for e-commerce, public opinion monitoring, advertising and other areas.

In this paper, Micro-blog user model construction method based on the ontology technology is discussed. First of all, according to each user's micro-blog content analysis, extract the keywords which represent the content information of each Micro-blog; then create user's eigenvectors in order to calculate the user's interest degree; finally match these keywords with the ontology library to create Micro-blog user interest model. The achievements of this paper will lay the foundation of personalized service based on Micro-blogs.

**Keywords:** Ontology, Micro-blog, User model, Interest Degree

# 绪论

本章主要描述了交通拥堵预警系统的背景、意义，研究分析了国内外相关课题的研究现状，进而提出了所要研究的内容及目标。

## 交通拥堵预警系统构建的背景及意义

近年来，随着城市经济高速发展，机动车保有量不断增加，交通拥堵现象日益严重。另一方面，随着商业5G部署的第一波浪潮的兴起，智慧交通的概念逐步变得清晰。依靠5G，人工智能，全球定位系统的协同合作，道路交通网将实现“车车入网”的智能化概念。

交通拥堵预警系统是智慧交通的重要组成部分，其依靠庞大的数据，高效的数据挖掘等技术，实现时间与空间上对未来道路交通情况的实时预测。对于交通拥堵的预警，不仅可以提醒即将经过拥堵路段的司机重新选择路线，从而防止拥堵加剧，而且可以帮助交警有重点地疏导交通，缓解拥堵趋势，从而预防拥堵发生。甚至可以与路网系统中的红绿灯系统融合，合理调整红绿灯状态，做到分流与疏通。

总的来说，交通拥堵预警可以帮助司机合理的规划出行路线，节省人们出行时间，有效地分配资源。也可以帮助交警出台合理的控制策略，缓解城市拥堵。

综上所述，本课题研究基于回归分析的交通拥堵系统具有现实意义。

## 模型研究现状及存在的问题

近年来，有关短时交通流预测的研究得到了国内外众多学者的高度关注，下面具体阐述研究现状以及存在的问题。

### 研究现状

应用于短时交通预测的模型主要可被归纳为统计模型，非线性预测模型与人工智能模型。

对于统计模型，Van Der Voort M[1]等人构建了结合自组织映射与时间序列模型的KARIMA模型，其通过将输入映射至最佳匹配单元，依据与该单元相连的可单独调整的ARIMA模型实现预测，凭借多个细分的时间序列模型，克服了单时间序列模型预测的局限性。对于非线性预测模型，宗春光[2]等人提出了一种基于相空间重构理论的局部预测方法。杨飞[3]提出了基于回声状态网络的交通流预测模型，并结合信号处理方法克服交通流中复杂的噪声成分的影响，取得了不错的预测效果。对于人工智能模型，谭娟[4]等人提出了基于深度学习的自编码网络，从无标签数据集中挖掘获取可表征数据深层特征的隐层参数，最后利用Softmax回归并进行多态预测。傅贵[5]等人深入研究了支持向量机回归，通过对算法本身的优化，使得预测误差小于基于卡尔曼滤波的预测方法。同样采用支持向量机回归，晏雨婵[6]等人通过粒子群优化算法，并结合模糊综合评价构建预测模型，取得了较高的准确度。Cheng[7]等人利用最大 Lyapunov 指数识别交通流混沌特征，结合相空间重构理论以及支持向量机回归，在预测准确性和及时性方面取得了更好的性能。

### 微博存在的问题

虽然，近年来微博得到了空前的发展。相比于传统博客，微博传播模式更加便捷，更新的频率更高。作为新兴的媒体，目前仍存在很多问题。

（1）很多用户感兴趣的有用信息，往往被迅速湮没。

（2）微博信息过于简单，微博之间的联系松散、逻辑关系复杂，容易引起误解。

（3）微博监管困难，对于敏感信息传播的预测和监控缺乏有效手段。

### 微博用户模型构建的难点

虽然用户建模技术已较为成熟，但针对微博这一特殊的平台，仍然存在了以下一些技术难点：

（1）对微博信息收集时，如何能够获取到大量有效的数据。

（2）微博信息短小精悍，对用户模型构建的准确性影响较大。

## 本文研究内容及目标

本文针对微博中存在的问题，试图对用户的微博内容进行分析，提取用户的兴趣，并建立微博用户模型，为微博信息推荐、舆情监控、微博营销等提供技术支持。

### 研究内容

本文研究基于本体的微博用户模型构建方法，具体研究内容有一下几个方面。

（1）领域本体构建；

（2）用户微博收集；

（3）微博内容分析；

（4）兴趣主题提取；

（5）用户模型构建。

### 研究目标

针对本文的研究内容，制定了以下几项指标：

（1）自动对搜集到的所有用户（实验10个以上）的所有微博（30条以上）进行分词；

（2）自动统计每个用户的关键词词频；

（3）合理计算每个用户模型中的兴趣度；

（4）根据已有的本体库建立用户模型；

（5）开发系统原型，验证提出的方法。

## 本文组织结构

整篇论文分为五章。

第一章介绍了研究背景、研究意义，分析了用户模型研究现状以及存在的问题和难点，并提出了本文的研究内容以及研究目标。

第二章主要介绍了本体的基本概念，并提出了基于维基百科的本体库构建方法。

第三章首先介绍了用户模型的基本概念及其表示方法，其次着重介绍了微博用户模型的构建方法，主要分为特征词提取、兴趣度计算和用户模型的生成。

第四章主要描述了微博用户模型构建系统设计与实现，展示了系统整体设计、数据库设计和各功能模块设计的内容。

第五章对全文进行了总结，归纳了本文的主要工作与创新点，并指出了需要进一步研究的问题。

# 基于维基百科的本体构建方法

本章具体描述了基于维基百科的本体构建方法：介绍了本体的基本概念，并引出本文所使用的基于维基百科的本体构建方法。

## 本体概述

本节介绍了本体的基本概念以及目前研究学者常用的四种构建方法。

### 本体的基本概念

在计算机领域，1991年开始，研究者们对本体做了多次说明，它表示的含义也更加清晰明确，现在人们一般认为本体论是对概念化对象的明确表示和描述[14]。随着研究者们对本体研究的不断完善，本体的定义有很多种，以下是几种比较有代表性的定义。

1991年，Neches[15]等给出了构成相关领域词汇的基本术语、关系，以及这些词汇外延的规则。1996年，Swartout[16]提出本体是一个知识库结构中术语集合，该结构中的术语是按照继承关系组织起来的，强调了本体中术语(Terms)的重要性。

1993年，Gruber[17]提出本体是概念模型明确的规范说明。1997年，Borst[18]提出本体是共享概念模型的形式化规范说明。1998年，Studer[19]等对上述两个定义进行了深入研究，认为本体是“共享概念模型的明确的形式化规范说明”，它有以下四个方面的含义：

（1）本体是一个概念模型(Conceptualization)，它是指通过抽象客观世界中一些现象的相关概念得到的模型。它表现的含义独立于具体的环境状态；

（2）本体的明确性(Explicit)，本体包含的概念和概念之间的约束都应该有明确的定义；

（3）本体是形式化的(Formal)，意思是本体应该是计算机可处理的；

（4）本体是可共享的(Sharable)，本体中概念、关系、属性的描述是基于标准的、规范的、能被共享的。

本体的定义多种多样，其核心都一样，把本体作为一种描述资源的手段，为不同的主体进行知识交流提供语义基础。在计算机领域中，对本体的研究主要是如何实现这种统一标准，以及本体的构建方法、本体描述语言、本体的管理和本体的应用等内容。

根据本体的语义特性，本体可以应用于不同的领域，致力于提高服务的联想能力和准确性。

总而言之，尽管本体的定义方式多种多样，但本体所包含的基本要素：概念、概念之间的关系等。

### 本体构建方法

目前，本体构建成功的案例很多，根据不同的领域，构建的方法也不一样，现在还没有构建本体的标准。许多研究人员根据经验总结出来了一些方法，1995年，Gruber提出构建本体的五条规则如下：

（1）明确性和客观性：本体应该是背景独立的、客观的，能反映社会真实情况，满足可计算性，具有明确的、客观的形式化语义；

（2）完整性：给出的定义应该是完整的，能表达特定属于的含义；

（3）一致性：只是推理产生的结论与属于本身的含义不产生矛盾；

（4）可扩展性：在扩展本体功能的时候，可以自由添加新的术语而对已有本体的结构和内容不做修改；

（5）最少约束：在满足可能的知识共享需求的基础上本体的约定应该最小。它可以通过只定义通讯所需的词汇或者定义约束最弱的公理来保证。

目前比较普遍的构造特定领域的本体，一般都需要相关领域专家的参与。以下列举了一些在项目实践过程中形成的方法：

IDEF-5方法：IDEF(ICAM Definition Languages)方法是上世纪七十年代由美国空军发明的。在1981年针对集成计算机辅助制造(Integrated Computer Aided Manufacturing，简称ICAM)项目中用于描述企业内部运作的建模方法。最初该方法只是应用于制造业，经过改造后，适用于软件开发。目前已经形成了一系列方法。包括IDEF1X和IDEF0到IDEF14共16套方法，每一套方法都通过建模程序获取某特定类型信息，其中IDEF-5是本体描述获取语言。

骨架法(Skeletal Methodology)：该方法是由爱丁堡大学人工智能应用研究所开发企业建模过程中总结出来的。

评价法：该方法由多伦多大学企业集成实验室，在开发虚拟企业本体工程项目时总结出来的。通过建立制定知识的逻辑模型，用一阶逻辑构造形式化的模型，包括企业设计本体、工程本体、计划本体和服务本体。

七步法：该方法是斯坦福大学医学院提出的基于Protégé本体构建工具的一种领域本体构建方法。一共包括七个步骤：

（1）确定只是本体的专业领域和范畴；

（2）考察复用现有只是本体的可能性；

（3）列出本题中的重要术语；

（4）定义类和类的层次体系；

（5）定义类的属性；

（6）定义类的分面(Facets)；

（7）创建本体实例。

## 基于维基百科的本体构建

以上传统的构建本体的方法仍然存在许多弊端，下面就针对这些不足进行分析阐述，并提出本文所使用的基于维基百科的本体构建方法的优势所在。

### 维基百科简介

维基百科(Wikipedia)是一个自由、免费、内容开放的网络百科全书，参与者来自世界各地。这个站点使用Wiki，这意味着任何人都可以编辑维基百科中的任何文章及条目。

维基百科是一个基于Wiki技术的全球性多语言百科全书协作计划，同时也是一部用不同语言写成的网络百科全书，其目标及宗旨是为全人类提供自由的百科全书──用他们所选择的语言来书写而成的，是一个动态的、可自由访问和编辑的全球知识体。

维基百科自2001年1月15日正式成立，由维基媒体基金会负责维持，其大部分页面都可以由任何人使用浏览器进行阅览和修改。因为维基用户的广泛参与共建、共享，维基百科也被称为创新2.0时代的百科全书、人民的百科全书。这本全球各国人民参与编写，自由、开放的在线百科全书也是知识社会条件下用户参与、大众创新、开放创新、协同创新的生动诠释。英语维基百科的普及也促成了其它计划，例如维基新闻、维基教科书等计划的产生，虽然也造成对这些所有人都可以编辑的内容准确性的争议，但如果所列出的来源可以被审察及确认，则其内容也会受到一定的肯定。维基百科中的所有文本以及大多数的图像和其他内容都是在GNU自由文档许可证下发布的，以确保内容的自由度及开放度。所有人在这里所写的文章都将遵循copyleft协议，所有内容都可以自由的分发和复制。

截至2013年1月，维基百科条目数第一的英文维基百科已有415万个条目，而全球所有282种语言的独立运作版本共突破2100万个条目，总登记用户也超越3200万人，而总编辑次数更是超越12亿次。大部分页面都可以由任何人使用浏览器进行阅览和修改，英文维基百科的普及也促成了其它计划。

Wiki一词来源于夏威夷语的“wee kee wee kee”，原本是“快点快点”的意思。在这里“WikiWiki”指一种超文本系统。这种超文本系统支持面向社群的协作式写作，同时也包括一组支持这种写作的辅助工具。

### 传统方式构建本体存在的问题

本体的构建是一个系统性工程，由于没有统一的构建原则、方法，到目前为止，本体工程仍处于相对不成熟的阶段，整个建设过程在以下几个方面还存在很多问题[20]。

（1）构建方法

7种构建方法都是从具体领域本体的开发中总结出来的，应用领域很有限，大多数方法的细节比较粗，相关技术比较少，没有一种方法完全按照生命周期法进行开发。另外，由于没有统一的构建原则作为指导，整个本体的构建过程难以进行规范的管理。

（2）可扩展性

随着领域的不断发展、变化，必然会有更多的领域相关概念和关系引入到核心本体中，需要通过知识的进一步获取、概念的进一步扩充或更改等方式，不断改进和扩展领域本体。但是目前本体的维护和扩展问题还没有得到很好的研究和支持。

（3）共享和重用

领域本体构建的目的是为不同系统提供彼此交流的语义基础。目前，为减少构建本体的工作量，多数的研究均利用现有术语丰富的叙词表作为构建工作的起点。而叙词表到本体的转换还没有一个统一的标准，在转换的过程中各项目使用的描述语言以及描述广度和深度不尽相同，给今后本体之间语义互操作和重用造成了困难。

### 基于维基百科的本体构建

维基百科全书于2001年1月投入运行，到2013年1月，英文版的维基百科全书就包含了415万多个条目，全部条目超过了2100万条，具有内容相互独立的282种语言版本。维基百科全书具有与大英百科全书类似的写作风格，相近的准确性，但内容的丰富性已经超过了大英百科全书。其内容不仅包括大量的插图，还包含了大量的多媒体内容，时效性更是印刷版的大英百科全书无法比拟的。与传统百科全书相比，维基百科特点突出：

（1）首先，维基百科始终将自己定位为包含人类所有知识领域的百科全书，而不仅仅是一本词典、在线论坛或其他。

（2）其次，计划也是一个Wiki，允许大众广泛参与。维基百科是第—个使用Wiki系统进行百科全书编撰工作的协作计划。

（3）最后，维基百科是一部内容开放的百科全书，其内容允许任何第三方不受限制地复制、修改及再发布，广泛的参与性，使得维基百科的权威性、中立客观性得到了保证。它方便不同行业的人士寻找知识，而使用者也可以不断增加自己的知识从而充实自己。当然，因为一些特殊的原因，目前中文版和英文版和百科全书也有部分敏感词汇被禁用。

目前，维基百科定期免费提供各个语言版本的所有数据，放到网络上供人下载。其中，数据包中包括有page.sql文件和categorylinks.sql文件，前者记录了所有页面的基本信息，如：页面标题、命名空间、页面长度等，后者记录了各个页面标记的类目信息。[21][22]虽然维基百科网站也提供了树状形式的类目网络供人浏览，但是这个网页的数据存在如下缺陷：

（1）由人工维护，所以数据更新比较缓慢；

（2）迫于部分类目深度过深，所以这个网页把这些类目删除掉，如“生物分类树”；

（3）由于采用简单的深度遍历算法将图输出为树状结构，所以部分类目深度失衡，例如“自然科学”类目就没有子类目。

基于以上原因，我们编程实现了类目网络的自动生成。基于维基百科中文本体的构建主要包括以下几个步骤：

（1）中文繁简转换

目前中文存在两种书写系统——繁体中文与简体中文。一般来自台湾、香港、澳门的使用者使用正体中文（繁体中文），来自中国大陆、新加坡、马来西亚则使用简体中文。[23]作为一个全球华人共同创作的平台，中文维基百科发布的数据中，既有繁体形式的，也有简体形式的，甚至很多在同一篇文章中繁简夹杂。这给我们的抽取工作带来很大的不便：一方面，文本繁简混杂的问题使得我们不能用现有的基于单一文字模式的中文信息处理工具直接分析文本；另一方面，繁简夹杂必然使得我们的研究成果不能很好地得到利用。所以，我们在利用维基百科所提供的简繁对应词表基础上，借鉴MediaWiki 1.4的繁简转换功能的“用字模式”，实现了繁简转换功能。

（2）类目网络清理

在维基页面分类系统中，存在着若干为了方便管理而添加的元类目，例如：“维基百科站务”。[24]因为这些类目所含语义信息较少，所以我们必须清理这些类目。我们剔除所有包含以下关键字的类目：维基，列表，模板，维基人，专题，分类，条目，小作品。另外，从数据库自动生成的类目网络中存在一些孤立点，我们将此类类目也全部清除。在清理之前，中文维基类目之间的直接连接数为21776个，清理后的直接连接关系总数14009个。

（3）识别父子关系

首先界定两个概念：下位词与“Is-a”关系（父子关系）[25][26]。语言学家Fromkin和Rodman[27]认为，下位词是一个一般化词语具体化之后的相关词语集合。例如，深红色，朱红，绯红色都是红色的下位词，而红色就是它们的上位词。同时，红色又是颜色的下位词。因此，下位关系也就是一般化术语（如多边形）和它的具体化实例（如三角形）之间的关系。在计算机科学中，常常将此关系称为Is-a关系[28]。例如，用“红色is a颜色”来描述红色和颜色之间的下位关系。在知识表示和面向对象编程与设计中，在A is a B中，Is-a表示类A是类B的子类，即B是A的父类。换言之，"A is a B"通常意味着概念A是概念B的具体化，概念B是概念A的一般化。举例而言，“水果”是“苹果”、“桔子”、“芒果”等概念的一般化。我们可以说，“苹果is a水果”。下文对这些类目之间的Is-a和Not Is-a关系进行自动识别。

经过以上步骤可以建立一棵基于维基百科的概念知识树，这就是我们维基百科本体知识树的原型。

## 本章小结

本体库的构建是微博用户模型构建的基础，基于所构建完成的类目网络结构，可以生成所有类目的树形结构本体，进而为解决知识的共享和重用问题提供了新思路，也成为当前信息科学领域的研究热点之一。由于传统本体构建方法存在诸多问题，本文中利用维基百科这一新的互动开放的信息交流平台，构建出中文本体库，为后续进行微博用户模型的构建打下了基础。但本体库的完备性将对用户模型构建的准确性影响较大，因此对本体的研究工作仍任重而道远。

# 基于本体的微博用户模型构建方法

本章是全文的重点章节，全面阐述了本文的工作内容。首先简单介绍了用户模型的概念及其表示方法，随后将本文的构建过程完整的进行了描述，并将部分系统功能运行的结果作了展示。

## 用户模型概述

在基于内容的个性化推荐中，文本内容是主要的研究对象。自然语言所表示的文本内容需要转化为机器能够识别和可计算的模型才能进行进一步的研究。用户模型并不仅仅是对用户兴趣的准确描述，由于可计算性是它对用户模型的基本要求，也就是说，用户模型不是对用户个体的一般性简单描述，而是一种面向算法的，具有特定数据结构，形式化的用户兴趣描述，它是实现个性化服务的基础和核心。

根据Gerhard Fischer[29]的论述，可以认为广义上的用户模型有如下三种：

（1）用户头脑中的概念模型。这是用户头脑中关于计算机系统及其所应具有的功能的模型，表示了用户对计算机系统的理解和期望，该模型随着用户使用系统经验的增加而不断完善；

（2）设计者的用户模型。设计者头脑中关于用户的模型，是设计者对用户特征的描述，被设计者用来作为系统设计的基础；

（3）计算机系统的用户模型。它是由设计者在设计阶段依据设计者的用户模型用计算机软件构造的，在系统的运行过程中实现的。

由此可以看出，用户建模是这样一个过程：设计者根据用户概念模型调整设计者用户模型，将设计者用户模型用软件的方法转换为计算机系统的用户模型，用户建模的最终目标是计算机系统模型。狭义上的用户模型是指软件系统的用户模型，我们更关心的也是计算机所拥有的关于用户特征的模型。

本文的用户模型是一种对于用户兴趣内容特征的描述和表达，可以收集并提取用户的兴趣偏好，并与本体库进行匹配，进而更好地理解用户的需求和任务，实现个性化的推荐服务。

## 用户模型表示方法

用户模型表示是用户建模的基础，决定了用户模型反映用户信息的能力和可计算能力。本节将讨论用户模型表示的有关方法。

### 常用用户模型表示方法

用户模型表示方法有很多种，目前使用较为广泛的有关键词表示法、主题表示法和向量空间模型表示法等。

关键词表示法是以用户感兴趣的一组关键词来表示用户模型。例如{篮球，火箭，后卫}。关键词可以由用户自己设定，也可以通过用户的行为、他们在网上留下的信息及其它特征来获取。

主题表示法是关键词表示法的一种改进，这种表示法是以用户感兴趣的信息的主题来表示用户模型的。例如用户对体育、文化、科技感兴趣，则该用户的用户模型表示为{体育，文化，科技}。

向量空间模型表示法则是利用特征词和相应的权值向量来表示用户模型。其基本思想是根据用户感兴趣的文档中各个关键词的出现频率建立特征词及其相应的权值向量来表示用户模型。

虽然存在多种不同的方法来表示用户模型，但常用的用户模型表示方法普遍存在以下两大问题：

（1）缺乏统一标准。仅能被特定的系统所运用，不能在不同系统间实现共享。

（2）缺乏领域知识支持。大部分的表示方式只是对用户兴趣的简单罗列，没有考虑用户兴趣间的关联关系，难以表达用户兴趣的语义内容。

### 本体用户模型表示方法

基于本体的用户模型表示方法使用规范的结构模式描述用户兴趣，把用户兴趣与领域知识的语义概念层次相结合，具有强大的语义表达能力、兼容性和可扩展性，可以很好地解决常用用户模型存在的问题。

本文中使用的本体用户模型是一棵带权的结点树，如**错误!未找到引用源。**所示，图中两结点之间存在父子关系，比如Sports是Football的父节点。节点旁边的数字则是每个兴趣节点的兴趣度大小，也就是用户对这个兴趣的感兴趣程度。

……

……

0.17

0.17

0.58

0.61

0.22

0.81

0.78

0.34

图3‑1本体用户模型兴趣树

基于本体的用户模型表示方法主要有以下几个优势：

（1）充分描述用户兴趣的语义。

用户兴趣中的一个词条往往包含丰富的语义。该方法能描述该词条对应的用户兴趣的层次概念，结合领域本体，使兴趣表示带着丰富的语义。

（2）兼容性和可扩展性。

该方法具有很好的兼容性和可扩展性。根据该方法建立的模型可以方便地移植到其它系统中，也可随着应用的发展，对模型进行扩展。

（3）自适应性。

随着现实世界知识体系的变化以及用户兴趣的变化，该方法能够自动适应这些变化，准确表达用户的当前兴趣。

可以说，基于本体的用户模型表示方法是用户模型表示方法的发展方向，因此本文将实现基于本体的用户模型构建。

## 微博用户模型的构建方法

微博用户模型的构建主要包括特征词提取、兴趣度计算以及用户模型的生成。以下将分为三个小节来详细描述构建方法。

### 特征词提取

首先，我们将获取微博用户信息，理想的用户建模方法应该无需用户主动提供任何信息。自动用户建模就是根据用户在微博平台上注册留下的信息及用户发布在微博平台上的微博信息来构建用户模型。我们可以通过微博平台提供的API，获取到用户的昵称、密码、编号、姓名、关注数、粉丝数、发表微博的数量等个人基本信息以及每个用户在平台上发布过的所有微博信息内容来构建用户模型。

在中文信息处理领域，对中文自动分词的研究已较为成熟，经典的分词方法主要有最大向前匹配法、逐词遍历匹配法、最小向前匹配法等。本文未对分词方法进行深入研究，而是选择直接使用开源的中国科学院计算技术研究所研制的汉语词法分析系统ICTCLAS (Institute of Computing Technology, Chinese Lexical Analysis System)[31]，该分词系统分词准确率达到97%以上。本文实现的本体构建系统中使用ICTCLAS汉语分词系统官网提供的32位Windows操作系统下的JAVA版本的中文分词和词性标注。

计算机不具有人类的智能，不能像人类一样阅读微博后根据自身的知识和理解能力对微博内容产生理解。因此，在进行微博语义扩展之前首先要将微博转换成易被计算机理解和识别的结构形式。微博的表示要求能够准确有效的表达微博内容，并且还要易于计算机处理。

目前，典型的文本表示方法主要有：布尔模型（Boolean Model）、向量空间模型（VSM）、语言模型（Language Model）、潜在语义索引（LSI, Latent Semantic Indexing）[32]和概率检索模型（Probability Model）[33][34]。这些模型从不同的角度出发，使用不同的方法标注特征词权重和相似度计算等问题。

向量空间模型由Salton等人于20世纪70年代提出，并成功地应用于著名的SMART文本检索系统。VSM被广泛应用于文本分类、文本聚类、信息检索等领域。近年来，在文本挖掘领域向量空间模型已经成为最常用的文本表示方法。VSM是基于这样一个关键假设下提出的，即文档中各词条出现的先后顺序是无关紧要的，每个特征词对应特征空间的一维，他们每一维对于判定文档所属的类别所起的作用是相互独立的。因此，可以把一篇文档看成是一系列无序词条的集合，从文档中选取出n个特征词来表示文本就是形成一个n维向量空间。例如一篇文档中选取三个特征词t1、t2、t3，那么这篇文档就表示为。但是对于整个文档来说，每个特征词对文本的重要程度不同，因此，需要对每个特征词赋予一定的权重。一篇具有n个特征词的文档利用VSM表示方法就可以表示为公式（1）。

 （1）

其中，di是第i篇文档，tij表示第i篇文档的第j个关键词，wij是第i篇文档的第j个关键词权重。关于权重的计算方法有很多种，将在3.3.2兴趣度计算步骤中再作介绍。本文中微博文本表示采用VSM方法。

### 兴趣度计算

经过文本分词处理后，需要抽取一定数量的特征词作为向量的各维表示文本。然而，文本中每个特征词对文本主题内容的贡献度不一样，即每个特征词的权重不同，如何准确有效地计算特征词权重成为重要的研究点。在研究最初特征词的权重只有0或者1，如果该特征词在文本中出现过它的权重就设为1，否则设为0。这种方法完全没有体现出在文本中出现的特征词之间对文本主题内容贡献度的差异性，所以这种权重计算方法慢慢被更精确的基于词频统计的方法替代。

常用的权重计算方法有布尔函数、特征词频平方根、WIDF函数及TF-IDF法等。目前使用最为广泛也是本文中所用到的方法就是TF-IDF法，计算方法如公式（2）所示。

**错误!不能通过编辑域代码创建对象。** （2）

其中，**错误!不能通过编辑域代码创建对象。**表示在文本**错误!不能通过编辑域代码创建对象。**中第j个特征词**错误!不能通过编辑域代码创建对象。**出现的次数，N表示文本集中所有文本数，n表示文本集中含有特征词**错误!不能通过编辑域代码创建对象。**的文本数，**错误!不能通过编辑域代码创建对象。**表示文本**错误!不能通过编辑域代码创建对象。**中第j个特征词的权重。根式**错误!不能通过编辑域代码创建对象。**是归一化因子。

将每个特征词在某个用户发表的所有微博中出现的权重相加，就得到了该用户对于该特征词的兴趣度值。

### 用户模型的生成

本文中利用基于维基百科的中文本体自动向上扩展建立一个树状层次结构，其中最主要的关系为上下位关系，下一层的同义词集是其父节点的下位关系，反之亦然。上下层的关系也是包含与被包含的关系，下一层的节点包含于其父节点。我们手动建立了一个基于维基百科分类下的分类特征词本体库，共输入有4757条记录，分为财经、IT、健康、体育、旅游、教育、招聘、文化、军事九个大类。

当之前分析出的用户兴趣主题与我们所建立的本体库中的某个词匹配时，说明用户也对该兴趣主题的直接父节点以及祖先节点感兴趣，只不过对它们的感兴趣程度有所差别。基于这个思想，用户模型构建中会自动查找并更新兴趣主题的父亲节点，直至根节点。

通过以上步骤就可以构建出微博用户模型了，包括用户兴趣树和兴趣度两大模块。在用户模型中，用户的兴趣表示为一棵用户兴趣的本体子树，如**错误!未找到引用源。**所示。这棵本体子树包含的信息主要有：

（1）用户的兴趣主题

（2）用户不同的兴趣主题对应的兴趣度

（3）兴趣主题之间的层次结构关系

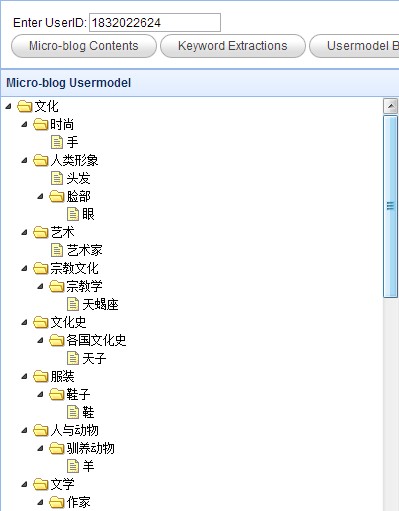
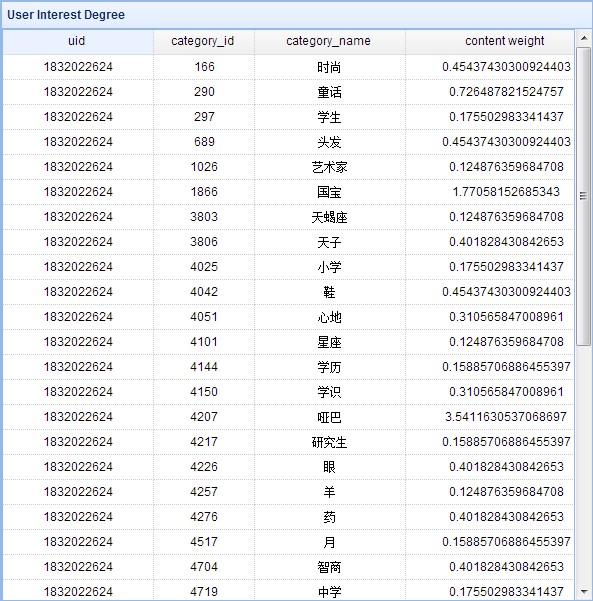


图3‑2用户模型中一个用户的兴趣树

某个用户对于其所有微博中的每个特征词的兴趣度值已经过计算得到，在用户模型构建中，也将这部分内容在界面上显示出来，如图‑3‑3所示。



图‑3‑3用户模型中一个用户对每个微博特征词的兴趣度

## 本章小结

本章主要介绍了用户模型的基本概念以及常用用户模型和本体用户模型的表示方法。另外，详细地描述了微博用户模型构建方法的三大步骤，也是本文的最主要内容和贡献。

常用的三种用户模型表示方法缺乏语义和统一的标准。而基于本体的用户模型表示方法不仅能充分描述用户兴趣的语义，还具有兼容性和可扩展性，很好地解决了常用用户模型表示方法存在的问题，是用户模型表示方法的发展方向，因此本文将实现基于本体的用户模型构建。

本文采用基于词频统计的分词方法进行微博文本内容的特征词提取；基于TF-IDF方法计算出微博特征词的权重，并相加得到其兴趣度；最后生成用户模型，显示出用户兴趣树以及用户对每个特征词的兴趣度。

# 微博用户模型构建系统设计与实现

本章主要介绍微博用户模型构建系统的设计与实现，包括系统整体设计、数据库设计以及系统内各功能模块的设计与界面展示。

## 系统整体设计

### 开发环境简介

本文中设计实现的用户模型评价系统使用的开发工具是Java开发平台Eclipse和关系型数据库MySQL。

Eclipse是一个开放源代码的软件开发项目，专注于为高度集成的工具开发提供一个全功能的、具有商业品质的工业平台。

基于Java的开发平台还有很多，比如JBuilder在以前比较流行，但它是收费的。它的优点在于可以拖拉窗体，比较适合桌面软件的开发。而Eclipse的优点在于它是开源的软件，不收费，并且拥有大量丰富的插件，现在大多数企业都用它来开发，是目前最流行的开发工具之一，所以本文选择Eclipse来进行微博用户模型的构建与展示。

MySQL是一个小型关系型数据库管理系统。与其他的大型数据库例如Oracle、DB2、SQL Server等相比，MySQL自有它的不足之处，如规模小、功能有限等，但是这丝毫也没有减少它受欢迎的程度。MySQL灵活、小巧、便捷，对于一般的个人使用者和中小型企业来说，MySQL提供的功能已经绰绰有余，而且MySQL也是开源软件。所以MySQL非常适合用于轻量级应用的开发，这也是本文选择MySQL作为数据库的原因。

### 系统整体架构

系统整体将分为五个步骤来构建：

（1）分类本体库构建：基于维基百科的分类手动建立一个本体库。

（2）微博特征词提取：收集某一用户所发表的所有微博内容，并对每条微博进行分词操作，提取出特征词，计算得到其权重。

（3）主题兴趣度计算：将每个特征词在所有微博中出现的权重相加，得到这个用户对每个微博特征词的兴趣度。

（4）用户模型生成：将特征词与分类本体库进行匹配，得到这个用户的所有兴趣并形成一棵兴趣树。

（5）系统设计与实现：在系统界面上显示用户的兴趣树以及该用户对每个微博特征词的兴趣度，构建完成一个微博用户的兴趣模型。

系统设计流程图如**错误!未找到引用源。**所示。

用户的微博内容收集

建立本体库

微博内容分词，提取特征词（关键词）

特征词权重的计算

特征词与本体库分类进行匹配

用户所有微博中各特征词权重加和，

得到兴趣度并显示

构建并显示用户模型

显示用户兴趣树

图4‑1系统流程图

## 数据库设计

数据库表的设计是依据本文第三章所描述的用户模型构建的三大步骤，创建了如下四张表格。

用户发表的所有微博的内容s\_weibo\_content（如**错误!未找到引用源。**所示），包括的字段信息有用户编号uid，该用户发表的微博编号weibo\_id，微博内容content等。

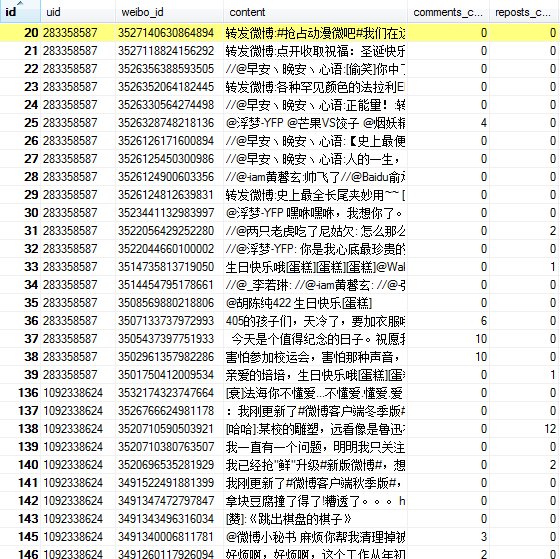


图4‑2微博内容数据表

对每条微博进行分词后所提取得到的特征词s\_feaword（如图4-3所示），包括的字段信息有用户发表的微博编号weibo\_id，从该微博提取到的特征词feaword，以及该词所占的权重weight。



图4‑3特征词数据表

手动创建的基于维基百科的分类本体库s\_category（如图4-4所示），包括的字段信息有分类序号ids，分类名称name，每个分类所对应的父节点序号parent\_id。



图4‑4分类本体库数据表

表4.1分类本体库数据表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **姓名** |  | **父节点** | **关系** |
| 1 | 陈寅恪 |  | 0 | 2 |
| 2 | 鲁迅 |  | 1 | 2 |
| 3 | 章太炎 |  | 2 | 3 |

最终构建完成的用户模型s\_user\_profile（如图4-5所示），包括的字段信息有用户编号uid，特征词的分类序号category\_id（即对应表s\_category中的分类序号ids），该特征词的内容权重contentweight等。

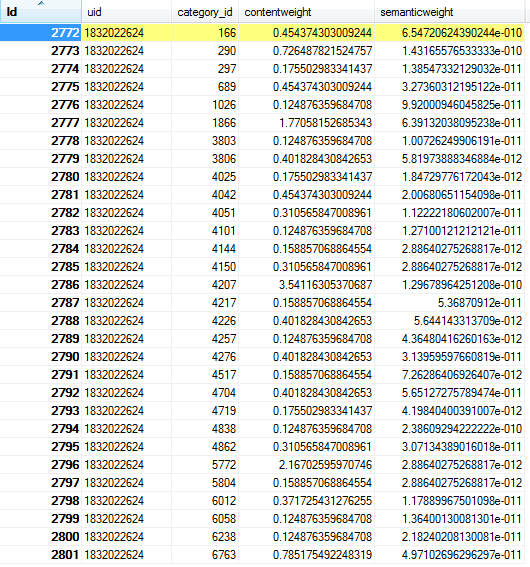


图4‑5用户模型数据表

## 功能模块设计

系统界面的功能模块主要分为本体的构建与显示、本体的构建与显示和用户模型构建与显示三个部分。

### 本体的构建与显示

基于维基百科的分类手动建立了4757条分类记录，初步形成了一个可作为研究基础的分类本体库，如**错误!未找到引用源。**所示。



图4‑6基于维基百科的分类本体库

### 用户基本数据显示

在系统界面上输入一个用户的ID号，就可以展示出该用户所发表的所有微博内容（如**错误!未找到引用源。**所示）以及从这些微博中提取得到的特征词和相应的权重（如**错误!未找到引用源。**所示）。



图4‑7用户发表的所有微博内容



图4‑8用户所有微博中提取的特征词及权重

### 用户模型构建与显示

构建完成的用户模型显示分为两部分，左边为所查询用户的兴趣树，右边为对应计算得到的用户对其所有特征词的感兴趣程度，如**错误!未找到引用源。**所示。



图4‑9用户模型的构建与显示

## 本章小结

本章主要介绍了微博用户模型构建系统的整体设计及各功能模块的展示，简单介绍了开发环境及系统的架构，并展示了部分数据表的内容及演示界面的布局、数据等。

# 总结与展望

本章对全文的主要工作和创新点作了总结，并提出需要进一步研究和改进之处。

## 本文总结

本文针对微博中存在的问题，对用户的微博内容自动进行分析，从而提取用户的兴趣，并建立微博用户模型，为微博信息推荐、舆情监控、微博营销等提供技术支持。

### 本文的主要工作

本文主要研究的是基于本体的微博用户模型构建方法，主要工作内容有以下几个方面：

（1）分类本体库构建：基于维基百科的分类手动建立了4757条分类记录，初步形成了一个可作为研究基础的分类本体库。

（2）微博特征词提取：收集某一用户所发表的所有微博内容，并对每条微博进行分词操作，提取出特征词，计算得到其权重。

（3）主题兴趣度计算：将每个特征词在所有微博中出现的权重相加，即得到了这个用户对每个微博特征词的兴趣度。

（4）用户模型生成：将特征词与之前建立的分类本体库进行匹配，得到这个用户的所有兴趣并形成一棵兴趣树。

（5）系统设计与实现：在系统界面上显示用户的兴趣树以及该用户对每个微博特征词的兴趣度，即构建完成了一个微博用户的兴趣模型。

### 本文的主要创新点

本文基于本体的微博用户模型构建方法创新点主要有以下两项：

（1）实现了利用微博信息分析用户兴趣方法。

（2）实现了利用本体构建用户模型的方法。

## 展望

虽然本文实现了基于本体的微博用户模型构建，但仍存在不少需要改进的地方：

（1）本体的完备性和微博的短文本对用户模型的准确性影响较大。由于手动建立的本体库有限，仅能作为初步研究工作的基础，但为了下一步研究的准确性更上一层楼，本体库的数据需要大量的增加，维护数据库的成本将会变得非常高。而短文本又是微博的一大特点，在140字以内的短文本中提取有效的特征词变得更加不易。

（2）本文对特征词所做的匹配仅限于其内容，但是中文中写法相同却有着截然不同含义的词汇有很多，因而会对用户模型的准确性带来考验。而若是加上语义的因素，难度又将增加不少。

# 致谢

XXX。

# 参考文献

1. Van Der Voort M, Dougherty M, Watson S. Combining Kohonen maps with ARIMA time series models to forecast traffic flow[J]. Transportation Research Part C: Emerging Technologies, 1996, 4(5): 307-318.
2. 宗春光,宋靖雁,任江涛,胡坚明.基于相空间重构的短时交通流预测研究[J].公路交通科技,2003(04):71-75.
3. 杨飞. 基于回声状态网络的交通流预测模型及其相关研究[D].北京邮电大学,2012.
4. 谭娟,王胜春.基于深度学习的交通拥堵预测模型研究[J].计算机应用研究,2015,32(10):2951-2954.
5. 傅贵,韩国强,逯峰,许子鑫.基于支持向量机回归的短时交通流预测模型[J].华南理工大学学报(自然科学版),2013,41(09):71-76.
6. 晏雨婵,白璘,武奇生,叶珍.基于多指标模糊综合评价的交通拥堵预测与评估[J/OL].计算机应用研究:1-7[2019-05-07].http://kns.cnki.net/kcms/detail/5

1.1196.TP.20181011.1259.032.html.

1. Cheng A, Jiang X, Li Y, et al. Multiple sources and multiple measures based traffic flow prediction using the chaos theory and support vector regression method[J]. Physica A: Statistical Mechanics and its Applications, 2017, 466: 422-434.

# 附录：部分源程序清单

//1. 分词、特征词提取

**publicvoid** Feaword()**throws** Exception

{

List<S\_user> userlist =findusers();

Map<String,Integer> globalmap= **new**HashMap<String,Integer>();//存放全部微博特征词

**int** weibo\_count=0;

**for**(**int** u=0;u<userlist.size();u++)

{

String uid=userlist.get(u).getUid();

List<S\_weibo\_content> weibolist=findByUid(uid);

**if**(weibolist==**null**||weibolist.size()==0)

{

}

**else**

{

weibo\_count=weibo\_count+weibolist.size();

**for**(**int** w=0;w<weibolist.size();w++)

{

S\_weibo\_content sweibo=weibolist.get(w);

//单条微博开始

Map<String,Integer> currentmap = **new** HashMap<String,Integer>();//存放单条微博特征词

SplitWord splitWord = SplitWord.*getInstance*();

splitWord.init();

**int** count=0;

String str=sweibo.getContent();

str=str.replaceAll(" ", "");

String result=**null**;

result=splitWord.split(str);

//System.out.println(result);

String[] rr=result.split(" ");

**for**(**int** i=0;i<rr.length;i++)

{

String[] rr1=rr[i].split("/");

**if**(rr1[1].equals("n"))

{

String noun=rr1[0];

count++;

**if**(currentmap.get(noun)==**null**)

{

currentmap.put(noun, **new** Integer(1));

}

**else**

{

Integer number = currentmap.get(noun);

number++;

currentmap.put(noun, **new**Integer(number));

}

}

}

Iterator iter = currentmap.entrySet().iterator();

**while** (iter.hasNext())

{

Map.Entry<String,Integer> entry = (Map.Entry<String,Integer>) iter.next();

String key=entry.getKey();

**if**(globalmap.get(key)==**null**)

{

globalmap.put(key, **new** Integer(1));

}

**else**

{

Integer number=globalmap.get(key);

number++;

globalmap.put(key, **new** Integer(number));

}

}

}//单条结束

}

System.*out*.println("用户"+uid+"的微博分词结束");

}

System.*out*.println("总微博数weibo\_count="+weibo\_count);

//2. 特征词权重计算

**for**(**int** u=0;u<userlist.size();u++)

{

String uid=userlist.get(u).getUid();

List<S\_weibo\_content> weibolist=findByUid(uid);

**if**(weibolist==**null**||weibolist.size()==0)

{

}

**else**

{

**for**(**int** w=0;w<weibolist.size();w++)

{

S\_weibo\_content sweibo=weibolist.get(w);

Map<String,Integer> currentmap = **new** HashMap<String,Integer>();

SplitWord splitWord = SplitWord.*getInstance*();

splitWord.init();

**int** count=0;

String str=sweibo.getContent();

str=str.replaceAll(" ", "");

String result=**null**;

result=splitWord.split(str);

System.*out*.println(result);

String[] rr=result.split(" ");

**for**(**int** i=0;i<rr.length;i++)

{

String[] rr1=rr[i].split("/");

**if**(rr1[1].equals("n"))

{

String noun=rr1[0];

count++;

**if**(currentmap.get(noun)==**null**)

{

currentmap.put(noun, **new** Integer(1));

}

**else**

{

Integer number = currentmap.get(noun);

number++;

currentmap.put(noun, **new** Integer(number));

}

}

}

Iterator iter2 = currentmap.entrySet().iterator();

**while** (iter2.hasNext())

{

Map.Entry<String,Integer> entry2 = (Map.Entry<String,Integer>) iter2.next();

String key=entry2.getKey();

**double** tf=(**double**)currentmap.get(key)/(**double**)count;

System.*out*.println("count="+count);

System.*out*.println("tf="+tf);

**double** idf=(**double**)weibo\_count/(**double**)globalmap.get(key);

idf=Math.*log10*(idf);

System.*out*.println("globalmap.get(key)="+globalmap.get(key));

System.*out*.println("idf="+idf);

**double** weight=tf\*idf;

S\_feaword ff=**new** S\_feaword();

ff.setSweibo(sweibo);

ff.setFeaword(key);

ff.setWeight(weight);

saveFeaword(ff);

}

}

}

}

//3.用户模型兴趣树的显示

<ul class="easyui-tree">

<s:iterator value="#request['menus']" id="c2">

<s:if test="#c2.parent\_id == 0">

<li data-options="state:'closed'">

<span><s:property value="#c2.name" /></span>

<ul>

<s:iterator value="#request['menus']" id="c3">

<s:if test="#c3.parent\_id == #c2.ids">

<li>

<span><s:property value="#c3.name" /></span>

<ul>

<s:iterator value="#request['menus']" id="c4">

<s:if test="#c4.parent\_id == #c3.ids">

<li>

<span><s:property value="#c4.name" /></span>

<ul>

<s:iterator value="#request['menus']" id="c5">

<s:if test="#c5.parent\_id == #c4.ids">

<li>

<s:property value="#c5.name" />

</li>

</s:if>

</s:iterator>

</ul>

</li>

</s:if>

</s:iterator>

</ul>

</li>

</s:if>

</s:iterator>

</ul>

</li>

</s:if>

</s:iterator>

</ul>

//4. 用户兴趣度计算

public void contentinterest() throws Exception

{

List<S\_user> userlist = findByuid("283358587");

//读取所有用户

if(userlist==null||userlist.size()==0)

{

}

else

{

//读取所有本体类别

List<S\_category> categorylist=ms\_categoryDao.getS\_categoryname();

if(categorylist==null||categorylist.size()==0)

{

}

else

{

for(int u=0;u<userlist.size();u++)

{

//读取某一个用户的所有微博

String uid=userlist.get(u).getUid();

List<S\_weibo\_content> weibolist=findByUid(uid);

if(weibolist==null||weibolist.size()==0)

{

}

else

{

for(int c=0;c<categorylist.size();c++)

{

//判断任意一条微博在每一个分类上的权重，如有则类别兴趣累计

Double wc=(double) 0;

Double ws=(double) 0;

System.out.println(weibolist.get(w).getWeibo\_id()); feaword=findBywid(s\_weibo\_content.getWeibo\_id());//找到微博w的所有特征词记录

List<S\_feaword> feaword=findBywfid(categorylist.get(c).getName(),uid);

if(feaword==null||feaword.size()==0)

{

}

else

{

for(int f=0;f<feaword.size();f++)

{

wc=wc+feaword.get(f).getWeight();

ws=ws+categorylist.get(c).getScd();

}

}

if(wc>0)

{

S\_s\_user\_profile user\_profile= new S\_s\_user\_profile();

user\_profile.setUser(userlist.get(u));

user\_profile.setCategory(categorylist.get(c));

user\_profile.setContentweight(wc);

user\_profile.setSemanticweight(ws);

try

{

s\_UserProfileDao.save(user\_profile);

}catch(Exception e)

{

e.printStackTrace();

}

}

}

}

}

}

}

System.out.println("用户完");

}

//5. 用户兴趣度显示

<table id=*"tt"*class=*"easyui-datagrid"*style="width:*auto*;">

<thead>

<tr>

<thdata-options=*"field:'uid',width:150"*align=*"center"*>uid</th>

<thdata-options=*"field:'s\_category.ids',width:100"*align=*"center"*>category\_id</th>

<thdata-options=*"field:'s\_category.name',width:150"*align=*"center"*>category\_name</th>

<thdata-options=*"field:'contentweight',width:200"*align=*"center"*>content weight</th>

</tr>

</thead>

<s:iteratorvalue=*"#request['plansearch']"*id=*"c2"*>

<tr>

<td><s:propertyvalue=*"#c2.uid"*/></td>

<td><s:propertyvalue=*"#c2.category\_id"*/></td>

<td><s:propertyvalue=*"#c2.category\_name"*/></td>

<td><s:propertyvalue=*"#c2.contentweight"*/></td>

</tr>

</s:iterator>

</table>