

SHANGHAIUNIVERSITY

毕业设计（论文）

**UNDERGRADUATEPROJECT(THESIS)**

**题目:区块链演示系统开发**

|  |  |
| --- | --- |
| **学院** | **计算机工程与科学学院** |
| **专业** | **计算机科学与技术** |
| **学号** | **14122006** |
| **学生姓名** | **梁尧** |
| **指导教师** | **丁广太** |
| **起讫日期** | **2018.01.29 – 2018.06.08** |

目录

[摘要 III](#_Toc460951271)

[ABSTRACT IV](#_Toc460951272)

[第1章 绪论 1](#_Toc460951273)

[§1.1数据产权保护的背景及意义 1](#_Toc460951274)

[§1.2数据产权保护现状及存在的问题 1](#_Toc460951275)

[§1.2.1研究现状 1](#_Toc460951276)

[§1.2.2数据产权保护存在的问题 2](#_Toc460951277)

[§1.2.3数据产权保护的难点 2](#_Toc460951278)

[§1.3本文研究内容及目标 2](#_Toc460951279)

[§1.3.1研究内容 3](#_Toc460951280)

[§1.3.2研究目标 3](#_Toc460951281)

[§1.4本文组织结构 3](#_Toc460951282)

[第2章 区块链 4](#_Toc460951283)

[§2.1区块链的起源 4](#_Toc460951284)

[§2.2区块链的概念 6](#_Toc460951287)

[§2.2.1区块链的定义 6](#_Toc460951288)

[§2.2.2区块链的关键词 7](#_Toc460951289)

[§2.2.3区块链的特性 7](#_Toc460951289)

[§2.2.4区块链的分类 7](#_Toc460951289)

[§2.3区块链基础技术 9](#_Toc460951291)

[§2.3.1哈希算法 6](#_Toc460951288)

[§2.3.1 Merkle树 6](#_Toc460951288)

[§2.3.2共识机制 6](#_Toc460951288)

[§2.4区块链原理 9](#_Toc460951291)

[§2.4.1区块链的技术原理 6](#_Toc460951288)

[§2.4.1区块链的运行原理 6](#_Toc460951288)

[§2.5区块链的发展进程 9](#_Toc460951291)

[§2.5.1区块链1.0 6](#_Toc460951288)

[§2.5.2区块链2.0 7](#_Toc460951289)

[§2.5.3区块链3.0 7](#_Toc460951289)

[§2.6本章小结 9](#_Toc460951291)

[第3章Hyperledger Fribic 10](#_Toc460951292)

[§3.1 Hyperledger Fribic的简介 10](#_Toc460951293)

[§3.2 Hyperledger Fribic的特性 10](#_Toc460951294)

[§3.2.1 模块化 11](#_Toc460951295)

[§3.2.2拥有权限机制的区块链网络 11](#_Toc460951296)

[§3.2.3智能合约 11](#_Toc460951296)

[§3.2.4新的事务架构 11](#_Toc460951296)

[§3.2.5私有性和保密性 11](#_Toc460951296)

[§3.2.6共识插件 11](#_Toc460951296)

[§3.3 Hyperledger Fribic的原理 12](#_Toc460951297)

[§3.3.1 Hyperledger Fribic的业务网络 12](#_Toc460951298)

[§3.3.2 Hyperledger Fribic的交易流程 14](#_Toc460951299)

[§3.4本章小结 16](#_Toc460951301)

[第4章Hyperledger Composer 10](#_Toc460951292)

[§4.1用户模型概述 10](#_Toc460951293)

[§4.2用户模型表示方法 10](#_Toc460951294)

[第5章数据保护业务网络的设计与实现 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6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)

XXX区块链演示系统开发XXX

# 摘要

XXX。

关键词：XXXX，XXX，XXX，XXX

XXXEnglish Title XXX

# 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

本章主要描述了微博用户模型构建的背景、意义，分析了相关课题国内外的研究现状，进而提出了本文所要研究的内容及目标。

## 数据产权保护的背景及意义

微博的数量和质量千变万化，各种海量、实时的数据信息已严重影响着用户接收信息的质量，进而影响着用户的生活质量。传统的人找信息和人找服务的模式已越来越难以满足用户的需求。如何为微博用户建立一个精准的用户模型，以便于之后为其推荐所关注的信息和感兴趣的内容，是各大移动电子商务网站进行个性化推荐的发展趋势。本课题研究在微博上基于本体的用户模型构建方法具有现实意义。

## 数据产权保护现状及存在的问题

近年来国内外学者对用户模型的研究做了大量的工作，而基于微博的用户模型构建也开始逐步成熟起来，下面具体阐述研究现状以及存在的问题。

### 研究现状

近年来，用户建模技术作为个性化服务中的基础，愈来愈受到重视，并逐渐地从个性化服务中独立出来，形成了专门的研究方向。研究人员逐渐意识到个性化服务质量的好坏不仅取决推荐技术或者检索技术，而且还取决于准确的用户模型。

国内的研究人员对用户模型构建和更新也展开了研究，例如大连理工大学林鸿飞和杨元生[1]根据用户提供的各类示例文档，通过考察特征、段落和类别的表达能力构建用户模型。南京大学多媒体技术研究所开发的个性化搜索引擎DOLTRI-Agent[2]采用一些相互关联的关键词组成用户模型，对每个关键词设置权值来表示用户对该关键词的感兴趣程度。国防大学的应晓敏[3]提出构建细粒度的基于关键词的用户建模方法，以更好体现出用户间的兴趣差异。国防科技大学的徐振宁[4]和李勇[5]构建了一个包括个性化领域本体的用户模型，跟踪记录用户在Web上的浏览和检索过程，从大量数据中统计、分析和计算出用户的个性化信息需求。

在国外，Fragoudis和Likothanassis[6]对几个典型的个性化服务系统LIRA[7]，Letizoa等采用的用户建模方法进行了综述和分析，指出用户建模在个性化服务系统中的重要地位。Pazzani和Binsusu[8]通过用户对浏览页面的标注获取用户感兴趣与不感兴趣的页面作为训练样本，而后计算单字的期望信息增益，选择期望信息增益大的128个单字构成用户模型。Chan[9]通过观察用户对页面中超链接的选择获取用户感兴趣与不感兴趣的页面作为训练样本，而后计算单字的期望互信息，选择期望互信息大的250个单字构成用户模型。Schwab[10]等通过观察用户对页面的选择获取用户感兴趣的页面作为训练样本，而后以出现在感兴趣页面中指定位置的单字构成用户模型。Adomavicious和Tuzhilin[11]采用数据挖掘方法对用户个体的访问记录进行挖掘，挖掘出来的关联规则以及用户登记的个人信息构成用户模型。

在微博的用户模型研究方面，近年来国内学者做了大量研究工作。例如广东社会主义学院的余伟[12]设计了一个基于本体的微博用户行为分析模型构架。北京邮电大学的赵岩露[13]等提出了基于微博用户兴趣模型的发现算法。而国外对Twitter的用户模型研究也有很多。

### 数据产权保护存在的问题

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。

### 数据产权保护的难点

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。

# 区块链

本章具体描述区块链的基本概念，介绍其特性、原理与发展现状，进而规划出其适合的应用场景。

## 区块链的起源

区块链是随着比特币诞生的比特币基础技术架构。所以要谈论区块链的诞生，我们必须从比特币的历史开始。

谈到比特币的起源，我们必须谈到一个神秘组织：密码Punk。这个组织是密码天才的松散联盟，比特币的创始、发展与密码Punk成员的贡献不无关系。密码Punk这个词语一部分来自于密码，在密码学中的含义是用于加密解密的算法；另一部分来自于Cyberpunk，它指的是那个时代流行的一个科幻类型。这样的组合有很微妙的意味，并散发着改变社会的激进理想。

密码Punk的观点是：现代社会对个人隐私和权力的侵蚀一直从未停止。他们对这一问题持续关注并一直不停地沟通，他们认为在数字时代，保护隐私对于维持社会的开放是至关重要的。这一理念在比特币中得到体现；去中心化、拥抱匿名机制、自由主义的原则。密码Punk是数字货币最早的发起者，在他们的邮件中，经常存在关于数字货币的讨论，并将部分想法付诸实现。比特币并不是数字货币的首次尝试。据统计，比特币诞生之前，有数十种失败的数字货币或支付。这些探索为比特币的诞生与发展提供了宝贵的可借鉴的经验。

2008年9月，从雷曼兄弟的倒闭开始，资本主义世界的金融危机率先在美国爆发，并迅速向世界蔓延。为应对这种危机，各国政府和中央银行决定采取史无前例的财政刺激方案和扩大货币政策，并向金融机构提供有限的紧急援助。这些措施同时也引起了社会各界的广泛质疑。2008年10月31日，在一个密码学邮件列表中，许多成员收到了自称是Satoshi Nakamoto的人的电子邮件，Satoshi Nakamoto给成员们传阅了自己撰写的一篇9页的白皮书，其中描述了一个新的货币系统。同年11月16日，Satoshi Nakamoto发布了比特币的Bata版本。2009年1月3日，Satoshi Nakamoto在位于芬兰赫尔辛基的一个小型服务器上挖出了比特币的第一个区块——创世区块，并获得第一批挖矿奖励——50个比特币，之后比特币在众人不看好的环境下，逐渐成长，价值不断翻倍。随着越来越多人的关注与进入，区块链逐渐与比特币解绑，成为一个新的技术与思想。

## 区块链的概念

本节介绍区块链的定义、特性与分类。

### 区块链的定义

区块链是集密码学、分布式网络、数据存储、共识算法等技术之大成。从广义上来讲，区块链技术利用了块链式数据结构来存储、验证数据，使用分布式节点共识算法来生成、更新数据，使用密码学加密的方式确保数据传输、访问的安全，使用由自动化脚本代码组成的智能合约来编写、操作数据的一种全新的分布式基础架构与计算范式。

### 区块链的关键词

1. 对等式网络（P2P）：

对等式网络是指允许整个系统作为有组织的集体操作的系统，允许单个节点与其他节点交互。拿比特币举例：比特币网络的构建方式是每个用户都在传播其他用户的任务。

1. 私钥：

非公开，拥有者需安全保管。通常是由随机算法生成的，是一个巨大的随机整数，256位、32字节。大小在1 ~ 0xFFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFE BAAE DCE6 AF48 A03B BFD2 5E8C D036 4141之间的数，都可被认为是一个合法的私钥。因此，除了随机方法外，使用特定算法从固定的输入获得到32字节输出的算法可以成为得到私钥的方法。

1. 公钥：

加密系统是加密手段的其中一种，私钥与公钥以一对多的关系存在，但是从公钥并不能推算出私钥的值，并且被用其中一个密钥加密了的数据，可以被另外一个相对应的密钥解密。该系统使得用户可以先公布一个公钥给其他人，然后其他人就可以发送加密后的信息给该用户，而无需预先交换密钥。

1. 数字签名：

将信息以双方事先约定好的Hash算法计算得到一个固定位数的字符串。在数学上保证：只要字符串中任意一位发生更改就会导致重新计算出的值与原先的值不相符。这样就保证了报文的不可更改性。

将该字符串值用发送者的密钥加密，并与原始消息一起发送给接收者，并将产生的信息称为数字签名。

### 区块链的特性

1. 去中心化

由于使用分布式核算和存储，没有集中的硬件或管理组织，任何节点的权利和义务是平等的，系统中的数据由整个系统中具有维护功能的节点维护。停止任意节点都不会影响系统的整体运作。

1. 去信任

系统中所有节点之间即便不存在信任也可以对等地进行交易，因为数据库和整个系统的运行和操作是开放的、透明的，在系统的规则和时间范围中，节点与节点间不存在欺骗的可能性。

1. 开放性

系统是开放的，除了交易中各方的私有信息被加密外，整个区块链中的数据对所有人都是公开的，所有人都可以通过公共的接口查询区块链中的数据或者开发相关的应用，因此整个系统的信息是高度透明化。

1. 自治性

区块链采用基于共识的规范和协议,这让整个系统中的全部节点能够在无需信任的环境中自由并且安全地交换数据，使得对“人类”的信任改成了对机器的信任，任何人为的干预不起决定性的作用。

1. 信息不可篡改

信息一旦经过验证并且添加到区块链中，这些信息就会被永久地存储起来，除非存在某个个人或组织能同时控制住世界各地区块链系统中超过51%的节点，否则少数节点对数据库的恶意修改是绝对无效的，这使得区块链的数据稳定性和可靠性非常高。

1. 匿名性

由于节点与节点间的数据交换遵守固定的算法，因此这些数据交互是不需要信任的(区块链中的编写好的程序规则会自行判断此项事务是否有效)，因此交易对手不需要通过公开身份让对方对自己产生任何信任，这对信用的累积非常有帮助。

### 区块链的分类

1. 私有链：

它指的是存在中心化组织控制的区块链。这里只是使用区块链的总账技术进行记账，这个中心化组织可以是一家公司，也可以是个体单位，这个组织独享这个区块链的写入的权限，这个链与其他的分布式存储方案没有较大差别。参与其中的节点只有用户自己而已，对数据的访问、使用自有有严格的权限管理。

特点：近乎是使用者拥有所有权，因此其中的数据不存在无法更改的特性，对于第三方使用者没有保障。商业应用场景是用作公司内部的审计。

1. 公有链：

公有的区块链是世界上最早的区块链，也是到目前为止使用最广泛的区块链。它指的是像比特币这样的完全去中心化的、不受任何组织控制的区块链。全世界任何个人或者组织都可以参与其中的交易，并且交易能够得到该区块链的有效的确认，所有人都可以参与该区块链的共识过程。共识过程中的参与者通过密码学技术和内置的经济激励来维护数据库的安全。

特点：完全开放、不受控制、依靠加密技术来保证安全。

1. 联盟链

它是指某个群体内部首先指定复数位预选节点作为为记账人，由所有的预选节点共同决定每个区块的生成，其他接入的节点可以参与其中进行交易，但并不过问记账的过程。另外，其他人可以通过此区块链开放的API接口进行有限定的查询。

参与区块链的节点是预选的，节点与节点之间有很好的网络连接。在该区块链上可以选择采用非工作量证明（PoW）的共识算法，举例来说：有100家金融组织之间建立了某个区块链，规则规定至少67个以上的组织同意才能达成共识。

特点：联盟链可以在节点与节点之间做到很好的连接，并且只需要很少的成本就能维持运营，提供迅速的交易处理和廉价的交易成本，它有良好的扩展性(但随着节点的增加，可扩展性会下降)，并且数据可以有一定的隐私性。联盟链也意味着该区块链的应用范围不会太广阔，并且缺少比特币的网络传播效应。

## 区块链基础技术

### 哈希算法

Hash算法是信息技术领域非常基础也非常重要的技术。它能任意长度的二进制值映射为较短的固定长度的二进制值，并且不同的明文很难映射为相同的 Hash值。

一个优秀的hash算法，将能实现：

（1）正向快速：给定明文和hash算法，在有限时间和有限资源内能计算出 hash 值。

（2）逆向困难：给定（若干）hash值，在有限时间内很难逆推出明文。

（3）输入敏感：原始输入信息修改一点信息，产生的hash值看起来应该都有很大不同。

（4）冲突避免：很难找到两段内容不同的明文，使得它们的 hash 值一致。

目前流行的Hash算法包括MD5、SHA-1和SHA-2。

MD5是Rivest于1991年对MD4的改进版本。它对输入仍以 512 位分组，其输出是128位。MD5 比MD4复杂，并且计算速度要慢一点，更安全一些。MD5已被证明不具备“强抗碰撞性”。

SHA是一个Hash函数族，由 NIST于 1993年发布第一个算法。目前知名的SHA-1在 1995 年面世，它的输出为长度160位的hash值，因此抗穷举性更好。SHA-1设计时基于和MD4相同原理，并且模仿了该算法。SHA-1已被证明不具备“强抗碰撞性”。

为了提高安全性，NIST还设计出了SHA-224、SHA-256、SHA-384，和 SHA-512算法（统称为SHA-2），跟 SHA-1 算法原理类似。SHA-3相关算法也已被提出。

目前，一般认为MD5和SHA1已经不够安全，推荐至少使用SHA2-256算法。

一般的，Hash算法都是算力敏感型，意味着计算资源是瓶颈，主频越高的 CPU进行Hash的速度也越快。也有一些 Hash算法不是算力敏感的，例如 scrypt，需要大量的内存资源，节点不能通过简单的增加更多CPU来获得hash 性能的提升。

### 椭圆曲线数字签名算法

椭圆曲线数字签名算法（ECDSA）是使用椭圆曲线密码（ECC）对数字签名算法（DSA）的模拟。ECDSA于1999年成为ANSI标准，并于2000年成为IEEE和NIST标准。它在1998年既已为ISO所接受，并且包含它的其他一些标准亦在ISO的考虑之中。与普通的离散对数问题（DLP）和大数分解问题（IFP）不同，椭圆曲线离散对数问题（ECDLP）没有亚指数时间的解决方法。因此椭圆曲线密码的单位比特强度要高于其他公钥体制。

### Merkle树

默克尔树是一种二叉树，由一个根节点、一组中间节点和一组叶节点组成。最下面的叶节点包含存储数据或其哈希值，每个中间节点是它的两个孩子节点内容的哈希值，根节点也是由它的两个子节点内容的哈希值组成。

Merkle树在分布式环境下验证、文件对比中应用较多。区块链系统采用二叉树型的Merkle树对这些交易进行归纳表示，同时生成该交易集合的数字签名。Merkle树支持快速地归纳和校验区块中交易的完整性与存在性。

### 共识机制

共识机制是通过特殊节点的投票，在很短的时间内完成对交易的验证和确认；对一笔交易，如果利益不相干的若干个节点能够达成共识，我们就可以认为全网对此也能够达成共识。

目前存在的共识机制有：

1. 工作量证明（PoW）：

工作量证明是矿工在处理交易数据的同时不断的进行哈希计算，求得一位前23位为0的哈希值，这个值成为nonce黄金数。当全网有一位矿工哈希出nonce时，他就会把自己打包的区块公布出去，其他节点收到区块验证区块后就会一致性认为这个区块接到了区块链上，就继续进行下一个区块的打包和哈希计算。

优点：

1.完全去中心化。

2.节点自由进出，容易实现。

3.破坏系统花费的成本巨大。

缺点：

1.对节点的性能网络环境要求高；

2.无法达成最终一致性。

3.浪费能源。

4. 容易产生分叉，需要等待多个确认。

1. 权益证明（PoS）：

权益证明，PoS是根据钱包里面货币的多少以及货币在钱包里存在的天数来合成一个单位（币天）。它根据币天的关系对计算机进行哈希计算降低了难度，降低了计算机的门槛，但是对计算机还是有一定要求的，它把钱包和区块链系统的一致性绑定在一起。谁的钱包里的币天数越大谁拥有记账权的概率就越大。但是它和POW机制一样解决问题的思想也导致了它与PoW拥有一样的缺点，也是牺牲了一部分的共识（同样分叉），而且需要等待多个确认。

优点：在一定程度上缩短了共识达成的时间，不再需要大量消耗能源挖矿。

缺点。

1. 任然需要挖矿，本质上没有解决商业应用的痛点。

2.所有的确认都只是一个概率上的表达，而不是一个确定性的事情，理论上有可能存在其他攻击影响。

3.没有最终一致性。

1. DPoS：

权益证明衍生机制，是基于PoS衍生出的更专业的解决方案，他是类似于董事会的投票机制，选举出n个记账节点，在节点中提案者提交的提案被这些记账节点投票决定谁是正确的。

优点：减少记账节点规模，属于弱中心化，效率提高。

缺点：牺牲了去中心化的概念，不适合公有链。

1. dBFT：

授权拜占庭容错机制，用权益来选出记账人，然后记账人之间通过拜占庭容错算法 达成共识。

优点：专业化的记账人可以容忍任何类型的错误记账由多人协同完成，每一个区块都有最终性，不会分叉算法的可靠性有 严格的数学证明

缺点：当三分之一或以上记账人停止工作后，系统将无法提供服务当三分之一或以上记账人联合作恶，且其他所有的记账人恰好分割为两个网络孤岛时，恶意记账人可以使系统出现分叉，但是会留下密码学证据

1. Paxos：

Paxos算法中将节点分为三种类型：

proposer：提出一个提案，等待大家批准为结案。往往是客户端担任该角色

acceptor：负责对提案进行投票。往往是服务端担任该角色

learner：被告知结案结果，并与之统一，不参与投票过程。可能为客户端或服务端基本过程包括 proposer 提出提案，先争取大多数 acceptor 的支持，超过一半支持时，则发送结案结果给所有人进行确认。

一个潜在的问题是 proposer 在此过程中出现故障，可以通过超时机制来解决。极为凑巧的情况下，每次新的一轮提案的 proposer 都恰好故障，系统则永远无法达成一致（概率很小）。Paxos 能保证在超过50%的正常节点存在时，系统能达成共识。

1. Paft：

Raft算法是对Paxos算法的一种简单实现。它包括三种角色：leader、candiate 和 follower，其基本过程为：Leader 选举：每个 candidate 随机经过一定时间都会提出选举方案，最近阶段中得票最多者被选为 leader同步 log：leader 会找到系统中 log 最新的记录，并强制所有的 follower 来刷新到这个记录，这里的log指的是各种事件的发生记录。

1. Pool验证池：

基于传统的分布式一致性技术，加上数据验证机制。

优点：不需要代币也可以工作，在成熟的分布式一致性算法（Pasox、Raft）基础上，实现秒级共识验证。

缺点：去中心化程度不如bictoin；更适合多方参与的多中心商业模式。

以上共识机制各有优劣，应用于不同场景之中。

## 区块链的原理

### 技术原理

区块链技术的核心是所有当前参与的节点共同维护交易及数据库，它使交易基于密码学原理而不基于信任，使得任何达成一致的双方，能够直接进行支付交易，不需第三方的参与。

从技术上来讲，区块是一种记录交易的数据结构，反映了一笔交易的流向。系统中已经达成的交易的区块连接在一起形成了一条主链，所有参与计算的节点都记录了主链或主链的一部分。一个区块包含以下三部分：交易信息、前一个区块形成的哈希散列、随机数。交易信息是区块所承载的任务数据，具体包括交易双方的私钥、交易的数量、电子货币的数字签名等；前一个区块形成的哈希散列用来将区块连接起来，实现过往交易的顺序排列；随机数是交易达成的核心，所有矿工节点竞争计算随机数的答案，最快得到答案的节点生成一个新的区块，并广播到所有节点进行更新，如此完成一笔交易。

### 运行原理

Satoshi Nakamoto在其比特币白皮书中非常详细地介绍了区块链系统的建立过程：

（1）新的交易向全网所有节点广播

（2）每个节点把收到的交易都写到一个区块中

（3）每个节点都在新的区块上进行计算，寻找一个工作量证明解

（4）某个节点找到工作量证明解时，就把其所在的区块向全网进行广播

（5）其他节点收到广播的这个区块后对其进行验证，只有所有交易都被验证时有效的且未被使用之后，该区块才能被认可

（6）每个节点通过将此区块的哈希值作为父哈希值来进行下一个区块的计算，表示节点认可了此区块有效

一般情况下，一笔交易必须经过至少6次确认，才能最终在区块链上被承认是合法交易。若达到6次确认后要想修改记录，花费代价太大，得不偿失。

由此可以看出，区块链本质上是一个去中心化的巨大账本数据库，作为比特币的底层技术，区块链是由一串使用密码学相关联所产生的数据块组成，每一个数据块中包含了多次比特币网络有效确认的信息。随着交易不断产生，矿工不断验证交易创造新的区块来记录最新的交易，这个帐本会一直增长延长。这些区块按照时间顺序线性补充到原有的区块链上。每一个节点都有一份完整的已有区块链备份记录，而这些都是通过进行数据验证算法解密的矿工网络自动完成。区块链上保留有所有关于每个节点和节点上比特币余额的信息，这些信息也被记录在完整的区块链上。公共式区块链帐本完全对外公开，这意味着区块链信息可以通过特定地址在区块链浏览器上进行查询。

新区块的生成将奖励矿工新的电子货币，还可以通过设置交易费用来奖励挖矿这种提供算力的行为，系统通过这样的方式完成电子货币的发行，这也让矿工有利可图，成为了矿工挖矿的主要动机。

有了区块链之后，当一个用户想要进行历史交易的验证时，可以通过一系列基于密码学与数据结构学的运算追踪交易所属的区块，从而完成验证。此外，对于随机数答案难度的调整可以控制新区块的生成速度；私钥的保密性可以保证和实现匿名交易；对于历史交易数据的剪枝可以实现硬盘空间的回收：经过Satoshi Nakamoto的测算，经过完全剪枝的区块链数据一年只生成4.2MB的数据量。

## 区块链的发展进程

### 区块链1.0

数字货币和支付行为组成。特征包括：以区块为单位的链状数据块结构；全网共享账本；非对称加密；源代码开源，主要具备的是去中心化的数字货币和支付平台的功能;目标是为了去中心化，比如比特币。

### 区块链2.0

智能合约开发和应用。智能合约：区块链系统中的应用，是已编码的可自动运行的业务逻辑，通常有自己的代币和专用开发语言；DAPP，包含用户界面的应用，包括但不限于各种加密货币，如以太钱包；虚拟机，用于执行智能合约编译后的代码，虚拟机是图灵完备的。智能合约开始在区块链上应用，用机器合约指令代替人工操作，让一切变得更加透明，高效，没人有人为操作，干扰。比如以太坊上的艾希欧，就大大降低了融资成本。

### 区块链3.0

支持复杂的商业应用。以EOS及Var为首，满足复杂的商业应用，区块链3.0则进一步超越了经济领域，可用于实现全球范围内日趋自动化的物理资源和人力资产的分配，促进科学、健康、教育等领域的大规模协作。总的来说他扩展到的金融行业之外，涵盖社会生活的方方面面，最明显特点是不再依靠某个第三人或机构获得信任或建立信用，还有节约人力和时间成本，提升效率。

## 本章小结

通过本章的论述，我们系统性地介绍了区块链的由来、定义、特性、原理、分类等要素，根据以上信息，我们可以描绘出其可能的应用场景。由于其去中心化与去信任的特性，它可以应用到企业间建立贸易。由于其记录不可篡改，它也可以很好地应用于资产凭证证明业务中。

区块链并非万能，由于其强线上弱线下的特性，所以线下资源与区块链结合并非十分紧密。由此可见，区块链怎样与线下资源结合产生新的产业模式将会是区块链下一个发展方向。

# Hyperledger Fribic

本章是将介绍区块链开发平台之一的Hyperledger Fribic，介绍其概念，特性与原理，并阐述为何选用Hyperledger Fribic为技术框架开发数据产权保护网络。

## Hyperledger Fribic的简介

Linux基金会在2015年创立了Hyperledger项目，以推进跨行业区块链技术。 比起定义为为单一的区块链标准，Hyperledger更鼓励通过促进社区发展进而促进开发区块链技术的方式，鼓励开放式开发的知识产权和随着时间的推移采用关键标准。

Hyperledger Fabric是Hyperledger中的区块链项目之一。像其他区块链技术一样，它具有分类账（Ledger），使用智能合约，并且是参与者管理其事务的系统。

Hyperledger Fabric从其他一些区块链系统中解脱出来的地方在于它是私密的并且是被许可的。 Hyperledger Fabric网络的成员通过一个可信的成员资格服务提供商（MSP）注册，而不是一个允许未知身份参与网络的开放式无权限系统（需要像“工作证明”这样的协议来验证交易和保护网络） 。

Hyperledger Fabric还提供多种可插拔选项。总帐数据可以以多种格式存储，共识机制可以交换进出，并支持不同的用户服务提供商（MSP）。

Hyperledger Fabric还提供创建channel的能力，允许一组参与者创建单独的事务分类账。这一功能对那些彼此是竞争对手的参与者来说尤为重要。因为他们彼此可能不希望对方知道自己在网络中的每一笔交易。例如：他们可能需要向另一些参与者提供优惠价格，此时双方可以形成一个channel，此时这些参与者就有该channel的分类账的副本。

## Hyperledger Fribic的特性

Hyperledger Fribic是专为企业环境而设计的一款开源的企业级许可分布式账本技术（DLT）平台，与其他流行的分布式账本或区块链平台相比，可提供一些关键的差异化功能。

差异化的一个关键点在于，Hyperledger是在Linux基金会的基础上建立的，该基金会在开放治理下培育开源项目的历史悠久且非常成功，培养了强大的可持续社区和蓬勃发展的生态系统。 Hyperledger由不同的技术指导委员会管理，Hyperledger Fabric项目由多个组织的不同维护人员负责管理。它有一个开发社区，自从它最早的提交以来已经发展到超过35个组织和近200个开发者。

Fabric具有高度模块化和可配置的架构，可为各种行业用例（包括银行，金融，保险，医疗保健，人力资源，供应链乃至数字音乐交付）提供创新，多功能性和优化。

Fabric是第一个支持通用编程语言（如Java，Go和Node.js）而不是受限域特定语言（DSL）的智能合约的分布式账本平台。这意味着大多数企业已经拥有开发智能合约所需的技能，并且不需要额外的培训来学习新的语言或DSL。

Fabric平台拥有权限功能，与无权限的公开网络不同，Fabric平台中的参与者知晓彼此（非匿名），因此完全不受信任。这意味着虽然参与者可能不完全相互信任（例如，他们可能是同行业的竞争者），但是网络可以根据参与者之间存在的信任构建的治理模型进行操作，例如处理纠纷的法律协议或框架。

这些差异化设计特性的结合使Fabric成为当今性能较好的平台之一，无论是在事务处理还是交易确认延迟方面，它都能够实现交易和智能合约（Fabric称为“chaincode”）的隐私和机密性。

### 模块化

Hyperledger Fabric被专门设计为具有模块化架构。 无论是共识插件、身份管理协议插件（如LDAP或OpenID Connect）、密钥管理协议或加密库，Fabric都可以个性化配置。该平台的核心设计都是为满足企业用例需求的多样性而设计的。

站在Hyperledger Fabric整体层次上，Fabric由以下模块化组件组成：

1. 可插拔订购服务（Ordering Service）就事务顺序达成共识，然后将块广播给其他节点。
2. 会员服务提供商（Membership Service Provider）插件负责将网络中的实体与加密身份相关联。
3. 可选的点对点Gossip服务（peer-to-peer gossip service）通过向其他节点订购服务来传播块输出。
4. 智能合约（Chaincode）在容器环境（例如Docker）中运行以进行隔离。 它们可以用标准编程语言编写，但不能直接访问分类帐状态。
5. 分类账可以配置为支持各种DBMS。
6. 认可和验证策略插件实施，可以针对每个应用程序独立配置。

### 拥有权限机制的区块链网络

在一个没有权限的区块链中，几乎任何人都可以参与，每个参与者都是匿名的。在这样的背景下，除了在一定深度之前区块链的状态是不可变的之外，就没有信任。为了减轻这种信任缺失，无权限的区块链通常使用“挖矿”的本地加密货币或交易费用来提供经济激励，以抵消参与基于“工作证明”的拜占庭式容错共识形式的巨大成本。

另一方面，权限区块链在一组已知的、经过鉴别并经常审核的参与者之间进行区块链操作，该参与者在产生一定程度信任的治理模型下运行。经过许可的区块链提供了一种方法来保护具有共同目标但不完全相互信任的一组实体之间的交互。通过依赖参与者的身份，有权限的区块链可以使用更传统的碰撞容错（CFT）或拜占庭容错（BFT）共识协议，这些协议不需要昂贵的挖矿成本

此外，在这种有权限的情况下，参与者通过智能合约故意引入恶意代码的风险会降低。首先，参与者彼此了解，无论是提交应用事务、修改网络配置还是部署智能合约，所有行为都将记录在为网络和相关事务类型建立的认可政策之后的区块链中。根据治理模式的条款，可以很容易地识别破坏者的身份并及时处理。

### 智能合约

智能合约在Fabric中被称之为Chaincode。它的功能是可信的分布式应用程序，从区块链中获得其安全/信任以及节点之间潜在的共识。 这是区块链应用程序的业务逻辑。

有三个关键适用于在应用平台部署的智能合约：

1. 许多智能合约在网络中同时运行。
2. 智能合约可能会动态部署。
3. 应用程序代码应该被视为不可信，甚至可能是恶意的。

大多数现有的支持智能合约的区块链平台遵循订单-执行（Order-Execute）架构，其中共识协议：

1. 验证和排序事务然后将它们传播到所有对等节点。
2. 每个对等方按顺序执行这些事务。

在使用订单-执行（Order-Execute）架构运行的区块链中执行的智能合约必须是确定性的; 否则，可能永远不会达成共识。为解决非确定性问题，许多平台都要求智能合约以非标准或特定领域语言（如Solidity）编写，以便消除非确定性操作。 这增加了开发难度，因为它要求开发人员学习一种新语言来编写智能合约，同时也可能出现编程错误。

此外，由于所有事务都由所有节点顺序执行，因此性能和规模受到限制。 因为智能合约代码需要在系统中的每个节点上执行，所以要求采取复杂的措施来保护整个系统免受潜在的恶意合同侵害，以确保整个系统的弹性。

### 新的事务架构

Fabric为事务架构引入了名为执行-订单-验证（Execute-Order-Validate）的新架构。它通过将交易流程分为三个步骤来解决订单-执行（Order-Execute）模型面临的弹性、灵活性、可扩展性、性能和机密性挑战：

1. 执行事务并检查其正确性，从而认可它。
2. 通过（插件）共识协议订购事务。
3. 在将它们提交到分类帐之前，根据应用程序特定的认可政策验证事务。

这种设计与订单-执行（Order-Execute）范式截然不同，因为Fabric在订购达成最终协议之前执行事务。

在Fabric中，应用程序特定的认可策略指定哪些对等节点或其中有多少节点需要担保正确执行给定的智能合约。 因此，每个事务只需要由满足事务的认可策略所必需的对等节点的子集来执行（认可）。这允许并行执行提高系统的总体性能和规模。 第一阶段也消除了任何不确定性，因为在订购前可能会过滤出不一致的结果。

由于我们已经消除了非确定性，Fabric是第一个可以使用标准编程语言的区块链技术。

### 私有性和保密性

在一个公开的，无权限的区块链网络中，利用PoW的共识模型，事务在每个节点上执行。这意味着合同本身以及它们处理的事务数据都不存在保密性。 每个事务以及实现它的代码对于网络中的每个节点都是可见的。在这种情况下，我们已经对PoW提供的拜占庭容错共识的合同和数据进行了保密性交易。

为了解决用于满足企业用例需求的隐私和机密性问题，区块链平台采用了多种方法。

加密数据是提供机密性的一种方法，然而，在利用PoW达成共识的无权网络中，加密数据位于每个节点上。给定足够的时间和计算资源，加密可能被打破。对于许多企业用例来说，其信息可能受到损害的风险是不可接受的。

零知识证明（ZKP）是解决这个问题的另一个研究领域，但存在的问题是，目前计算ZKP需要相当多的时间和计算资源。

Hyperledger Fabric作为一个有权限的平台，通过其通道（Channel）架构实现了机密性。Fabric网络上的参与者可以在参与者的子集之间建立一个通道（Channel），这些参与者应该被授予对特定交易集的可见性。把它想象成一个网络覆盖。因此，只有那些参与通道（Channel）的节点才能访问智能合约（Chaincode）和数据事务，保护隐私和机密性。

### 共识插件

事务的排序被委托给模块化组件以实现共识，这在逻辑上与执行事务和维护分类账的对等方分离。由于共识是模块化的，因此其实施可以根据特定部署或解决方案的信任假设进行量身定制。 这种模块化架构使平台能够依靠成熟的CFT（碰撞容错）或BFT（拜占庭容错）排序工具包。

Fabric提供使用Kafka和Zookeeper实施的CFT订购服务。在未来，Fabric将提供一个用etcd / Raft实现的Raft一致性订购服务和一个完全分散的BFT订购服务。

共识模组不是相互排斥的。 Fabric网络可以有多种支持不同应用或应用需求的订购服务。

## Hyperledger Fribic 的原理

本节主要介绍Hyperledger Fribic的原理机制，探究其业务网络与交易流程。

### Hyperledger Fribic的业务网络

业务网络，也叫共识网络或区块链网络，由不同的节点构成。节点是区块链的通信实体，节点是一个逻辑概念，不同类型的节点可以运行在同一台物理服务器上。这些节点可能部署在云上面或者本地。可能来自不同的公司或者组织。在区块链网络中有两种类型的节点：Peer节点和Orderer节点。

### 

Peer节点：Chaincode部署在Peer节点上，它对账本进行读写操作。一个Peer节点可以充当多种角色，如背书者endorser，提交者committer。一个区块链网络中会有多个Peer节点。

Orderer节点：对交易进行排序，批量打包，生成区块，发给Peer节点。一个区块链网络中会有多个Orderer节点，它们共同提供排序服务。排序服务可以用多种不同方式实现，例如，从一个中心化的服务（如Solo）,到分布式协议（如Kafka）。

排序服务提供了通向客户端和Peer节点的共享通信通道。提供了包含交易的消息广播服务（broadcast、deliver）。客户端可以通过这个通道向所有的节点广播（broadcast）消息。通道可以向连接到该通道的节点投递(deliver)消息。

排序服务支持多通道，客户端和Peer节点可以连接到指定的通道（channel），并通过指定的通道发送和接收消息。多通道使得Peer节点可以基于应用访问控制策略来订阅任意数量的通道;也就是说，应用程序在指定Peer节点的子集中架设通道。这些peer组成提交到该通道交易的相关者集合，而且只有这些peer可以接收包含相关交易的区块，与其他交易完全隔离，实现数据隔离和保密。

此外，peers的子集将这些私有块提交到不同的账本上，允许它们保护这些私有交易，与其他peers子集的账本隔离开来。应用程序根据业务逻辑决定将交易发送到1个或多个通道。

### Hyperledger Fribic的交易流程

1. 客户端提交交易提案

客户端利用任意SDK提交交易提案（propose）。该提案是一个调用智能合约功能函数的请求，用来确认哪些数据可以读取或写入账本。

客户端把交易提案发送给一个或多个Peer节点，交易提案中包含本次交易要调用的合约标识、合约方法和参数信息以及客户端签名等。

SDK将交易提案打包为可识别的格式（如gRPC上的Protocolbuffer），并使用用户的加密凭证为该交易提案生成唯一的签名。

1. 背书节点模拟执行交易

背书节点endorser收到交易提案后，验证签名并确定提交者是否有权执行操作。背书节点将交易提案的参数作为输入，在当前状态Key-Value数据库上执行交易，生成包含执行返回值、读操作集合和写操作集合的交易结果（此时不会更新账本），这些值的集合、背书节点的签名和背书结果（YES / NO）作为提案的结果返回给客户端SDK，SDK解析这些信息判断是否应用于后续的交易。

1. 客户端把交易发送到共识服务

应用程序（SDK）验证背书节点签名，并比较各节点返回的提案结果，判断提案结果是否一致以及是否参照指定的背书策略执行。

客户端收到各个背书节点的应答后，打包到一起组成一个交易并签名，发送给Orderers。

1. 共识排序，生成新区块，提交交易

Orderers对接收到的交易进行共识排序，然后按照区块生成策略，将一批交易打包到一起，生成新的区块，调用deliver API投递消息，发送给提交节点。

提交节点收到区块后，会对区块中的每笔交易进行校验，检查交易依赖的输入输出是否符合当前区块链的状态，完成后将区块追加到本地的区块链，并修改Key-Value状态数据库。

## 本章小结

本章主要介绍了区块链开发平台之一的Hyperledger Fribic，通过介绍其概念、特性、原理等内容。我们可以看出，Hyperledger Fribic在不损失区块链本身特性的情况下，引入了权限控制与隐私、保密等属性。这些新的引入满足商业化应用场景的需求，因而Hyperledger Fribic是开发商业化区块链应用程序的首选之一。

# 数据保护业务网络的设计与实现

本章主要介绍数据保护业务网络的设计与实现，包括系统整体设计、数据库设计以及系统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

本文中设计实现数据保护业务网络使用的开发工具是Docker容器、Hyperledger Composer、非关系型数据库CouchDB、REST API接口LoopBack等工具。

1. Docker是一个开源的应用容器引擎，基于Go语言 并遵从Apache2.0协议开源。Docker可以让开发者打包他们的应用以及依赖包到一个轻量级、可移植的容器中，然后发布到任何流行的Linux机器上，也可以实现虚拟化。容器是完全使用沙箱机制，相互之间不会有任何接口，更重要的是容器性能开销极低。
2. Hyperledger Composer是一个广泛的、开放的开发工具集和框架，可以使开发区块链应用程序变得更容易。它的主要目标是加快内容开发的时间，并使开发者更容易将区块链应用程序与现有业务系统集成。 开发者可以使用Composer快速开发用例，并在几周内部署区块链解决方案。 Composer允许开发者为业务网络建模，并将现有系统和数据与区块链应用程序集成。
3. Apache CouchDB数据库，它类似于Redis，Cassandra和MongoDB，也是一个NoSQL数据库。 CouchDB将数据存储为非关系性的JSON文档。 这使得CouchDB的用户可以以与现实世界相似的方式来存储数据。
4. LoopBack是一个高度可扩展的开源Node.js框架，它使您能够创建动态的端到端REST API，而且只需少量编码或不需要编码。

### 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。

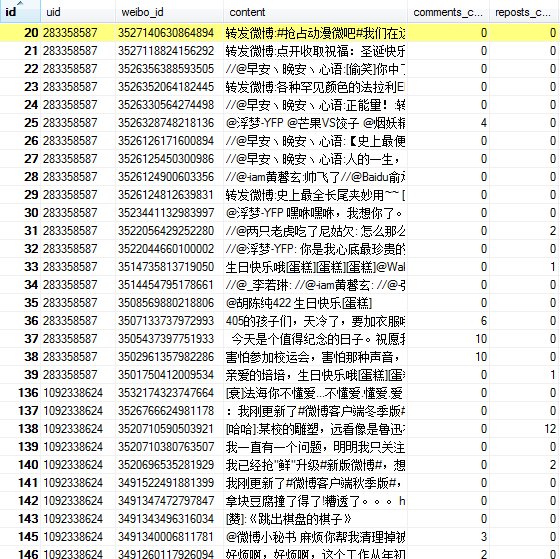


图4‑2微博内容数据表

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。

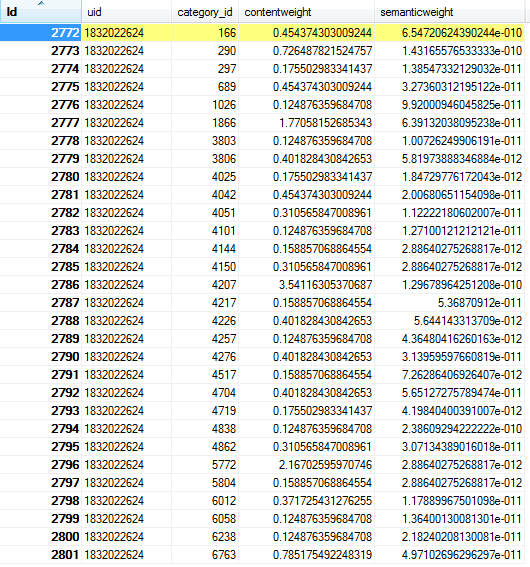


图4‑5用户模型数据表

## 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. 林鸿飞,杨元生.用户兴趣模型的表示和更新机制.计算机研究与发展，200239(7):838-842.
2. 胡学联,潘金贵,李俊,张灵玲. 一个个性化的信息搜集Agent的设计与实现.软件报,2001,12(7):1074-1079.
3. 应晓敏,面向Internet个性化服务的用户建模技术研究. 2003,国防科技大学:长沙
4. 徐振宁,张维明,陈文伟. 基于Ontology的智能信息检索. 计算机科学,20.28(6):21-26.
5. 李勇.智能信息检索中基于本体的个性化用户建模技术及应用. 2002,国防科技学:长沙.
6. FragoudisD.User Modeling in Information Discovery:An overview.InProceedings of Advanced Course on Artificial Intelligence,1999(ACAI99), July,1999, Greece.
7. Balabanovie M, Shoharn Y.Learning Information Retrieval Agents:Experiments with Auto-mated Web BrowsingIn:Proceedings of the AAAISpring Symposium Series on Information Gathering from Heterogeneous, Distributed Environments, March, 1995:13-18.
8. Lieberman H.Letizia. An Agent that Assists Web Browsing.In:Proceedings ofthe International Joint Conference on Artificial Intelligence, Montreal, August, 1995:924-929.
9. Chan P.K.A Non-Invasive Learning Approach to Building Web User Profiles in KDD-99Workshop on Web Usage Analysis and User Profiling, 1999, New York:ACMpress.
10. Schwab L,Kobsa A,and Koychev I. Learning about User from Observation[J].in AAAISpring Symposium on Adaptive User Interface, 2000, Standord,California:AAAI Press.
11. Adomavicius G and Tuzhilin A.Using Data Mining Methods to Build CustomerProfiles.IEEE Computer.Feb 2001:74-82.
12. 余伟. 基于本体的微博客用户行为模型研究[J]. 广东技术师范学院学报, 2010, 31(006): 27-30.
13. 赵岩露, 王晶, and 沈奇威. "基于特征分析的微博用户兴趣发现算法." 电信工程技术与标准化 25.11: 79-83.
14. Uschold M. and Gruninger M., Ontologies: Principle, Methods and Application. Knowledge Engineering Review, 1996, 11(2): p.93-155.
15. Neches R, et al. Enabling technology for knowledge sharing.AI Magazine, 1991, 12(3): p.36-56.
16. Swartout W. Ontologies, 1999. Intelligent Systems and their Applications, IEEE. Issue:1, p.18-19.
17. Gruber T.R.,A Translation Approach to Portable Ontology Specifications .knowledge acquisition, 1993, 5(2): p.199-221.
18. Borst W.N. Construction of Engineering Ontologies for Knowledge Sharing and Reuse. 1997, University of Twente: Enschede.
19. Studer R, Benjamins VR, and Fensel D. Knowledge Engineering: Principles and Methods. Data and Knowledge Engineering, 1998, 25(1-2): p.161-197.
20. 朱晓冰，寇雅楠基于维基技术的本体构建方法探讨图书馆学研究2009.1: p.55-58.
21. 于江生，俞士汝.中文概念词典的结构.中文信息学报.2002年第4期: p.64-68.
22. 由丽凭，杨翠.汉语框架语义知识库概述电脑开发与应用.2007年06期: p.72-74.
23. 张晶，姚建民，赵铁军，李生.基于WordNet和HowNet建设双语语义词典.高技术通讯.2001.12(2-3): p.43-46.
24. 侯汉清，薛春香.用于中文信息自动分类的《中图法》知识库的构建.中国图书馆学报.2005年第5期: p.67-70.
25. 罗志成，马费成，吴晓东，宋倩倩.从维基分类系统构建中文语义词典研究.信息系统学报.2008年02期: p.35-39.
26. 杨玲贤.基于ontology的教学资源知识库构建.计算机与现代化.2009年第11期: p.27-32.
27. Fromkin V, Rodman R. Introduction to Language. London: Holt, Rinehart and Winston, Inc., 1988, 12(1): p.10–16.
28. Brachman R, What IS-A is and isn't: An analysis of taxonomic links in semantic networks [J]. IEEE Computer, 1983, 16(10): p.30–36.
29. Fischer G. User modeling in human–computer interaction[J]. User modeling and user-adapted interaction, 2001, 11(1-2): 65-86.
30. 王金花. 一种利用本体关联度改进的TF-IDF特征词提取方法[D]. 河北大学, 2011.
31. 中国科学院计算技术研究所[EB/0L]. 中文自然语言处理开放平台. http://www.nlp.org.en/project/ project.php?proj\_id=6, 2005-2-2.
32. Scott Deerwester, Susan T Dumais, Furnas W George, et al. Indexing by latent semantic analysis[J]. Journal of the American Society for Information Science, Vol 41, 1990.
33. 王娟琴. 三种检索模型的比较分析研究—布尔模型、向量模型、概率模型[J]. 情报科学, 1998, 16(3): 225-231.
34. 景玉峰等. 概率检索模型.现代图书情报技术, 1987(1): 29-31.
35. 冀胜利, 李波. 基于SVM的中文文本分类算法[J]. 重庆工学院学报(自然科学), 2008, Vol.22 No.7: 84-87.
36. 崔争艳. 中文短文本分类的相关技术研究[D].
37. 埃克尔著, 陈昊鹏译. Java编程思想[M]. 第4版. 北京: 机械工业出版社, 2007.
38. 李刚. 疯狂Java讲义[M]. 第2版. 北京: 电子工业出版社, 2008.
39. 李刚. 轻量级Java EE企业应用实战:Struts2＋Spring3＋Hibernate整合开发[M]. 第3版. 北京: 电子工业出版社, 2011.
40. Dou Shen, Jian-Tao Sun, Qiang Yang, et al. Latent Friend Mining from Blog Data[C]. Sixth International Conference on Data Mining, 2006: 552-561.
41. Katarzyna Musiał, Przemysław Kazienko, and Tomasz Kajdanowicz. Social Recommendations within the Multimedia Sharing Systems[C]. First World Summit on the Knowledge Society, 2008: 364-372.

# 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>