

SHANGHAIUNIVERSITY

毕业设计（论文）

**UNDERGRADUATEPROJECT(THESIS)**

**题目:** **Android手机上的人脸实时检测APP的开发**

|  |  |
| --- | --- |
| **学院** | **计算机工程与科学学院** |
| **专业** | **计算机科学与技术** |
| **学号** | **14122556** |
| **学生姓名** | **雷佳霖** |
| **指导教师** | **支小莉** |
| **起讫日期** | **2018.01.29 – 2018.06.08** |

目录

[摘要 III](#_Toc515023970)

[ABSTRACT IV](#_Toc515023971)

[第1章 绪论 1](#_Toc515023972)

[§1.1 微博用户模型构建的背景及意义 1](#_Toc515023973)

[§1.2 用户模型研究现状及存在的问题 1](#_Toc515023974)

[§1.2.1 研究现状 1](#_Toc515023975)

[§1.2.2 微博存在的问题 2](#_Toc515023976)

[§1.2.3 微博用户模型构建的难点 2](#_Toc515023977)

[§1.3 本文研究内容及目标 2](#_Toc515023978)

[§1.3.1 研究内容 2](#_Toc515023979)

[§1.3.2 研究目标 3](#_Toc515023980)

[§1.4 本文组织结构 3](#_Toc515023981)

[第2章 基于PRN的目标检测方法 5](#_Toc515023982)

[§2.1 本体概述 5](#_Toc515023983)

[§2.1.1 本体的基本概念 5](#_Toc515023984)

[§2.1.2 本体构建方法 6](#_Toc515023985)

[§2.2 基于Region Proposal的目标检测模型 7](#_Toc515023986)

[§2.2.1 Faster-RCNN系列模型结构概述 8](#_Toc515023987)

[§2.2.2 传统方式构建本体存在的问题 8](#_Toc515023988)

[§2.2.3 基于维基百科的本体构建 9](#_Toc515023989)

[§2.3 基于独立神经网络的目标检测模型 11](#_Toc515023990)

[§2.3.1 SSD系列模型结构概述 11](#_Toc515023991)

[§2.3.2 YOLO系列模型结构概述 11](#_Toc515023992)

[§2.4 本章小结 11](#_Toc515023993)

[第3章 基于YOLO的目标检测方法 12](#_Toc515023994)

[§3.1 YOLO模型概述 12](#_Toc515023995)

[§3.2 YOLO系列模型结构发展与变化 13](#_Toc515023996)

[§3.2.1 YOLO v1模型结构概述 13](#_Toc515023997)

[§3.2.2 YOLO v2模型结构概述 15](#_Toc515023998)

[§3.2.3 本体用户模型表示方法 21](#_Toc515023999)

[§3.3 微博用户模型的构建方法 22](#_Toc515024000)

[§3.3.1 特征词提取 23](#_Toc515024001)

[§3.3.2 兴趣度计算 24](#_Toc515024002)

[§3.3.3 用户模型的生成 24](#_Toc515024003)

[§3.4 本章小结 26](#_Toc515024004)

[第4章 微博用户模型构建系统设计与实现 27](#_Toc515024005)

[§4.1 系统整体设计 27](#_Toc515024006)

[§4.1.1 开发环境简介 27](#_Toc515024007)

[§4.1.2 系统整体架构 27](#_Toc515024008)

[§4.2 数据库设计 28](#_Toc515024009)

[§4.3 功能模块设计 32](#_Toc515024010)

[§4.3.1 本体的构建与显示 32](#_Toc515024011)

[§4.3.2 用户基本数据显示 33](#_Toc515024012)

[§4.3.3 用户模型构建与显示 34](#_Toc515024013)

[§4.4 本章小结 35](#_Toc515024014)

[第5章 总结与展望 36](#_Toc515024015)

[§5.1 本文总结 36](#_Toc515024016)

[§5.1.1 本文的主要工作 36](#_Toc515024017)

[§5.1.2 本文的主要创新点 36](#_Toc515024018)

[§5.2 展望 36](#_Toc515024019)

[致谢 38](#_Toc515024020)

[参考文献 39](#_Toc515024021)

[附录：部分源程序清单 41](#_Toc515024022)

Android手机上的人脸实时检测APP的开发

# 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。

随着自动驾驶汽车，智能视频监控，面部检测和各种人员计数应用的兴起，快速而准确的物体检测系统的需求不断增长。这些系统不仅要识别和分类图像中的每个对象，还要通过在其周围绘制适当的边界框来定位每个对象，与传统计算机视觉的前身——图像分类相比，目标检测对于现存的传统机器学习算法来说计算量更庞大，任务更艰巨。

## 微博用户模型构建的背景及意义

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。

## 用户模型研究现状及存在的问题

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。

### 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及研究意义，阐述了YOLO系列检测方法在目标检测领域的现状以及存在的问题和难点，章节末尾引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。

第二章主要介绍了目标检测领域主要使用的几个著名模型，如SSD系列、Faster-RNN系列等。同时解释了目标检测领域的基本术语，汇总考察了适用于该领域的数据集。

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。

第三章介绍了YOLO系列YOLO v1、v2等网络模型的基本结构，并分析对比了该领域现存的几个著名模型与YOLO v2的差异。

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。

第四章主要描述了YOLO v2模型在Android端实现的整个应用设计方式与构建方式，以人脸检测为例进行，实现了深度学习模型在Android端的落地，并展示了数据集收集过程和Android端优化等各其他模块的内容。

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PRN的目标检测方法

主要介绍了目标检测的基本概念以及该领域常用的基本方法，简单介绍了YOLO系列、SSD系列和基于RPN的R-CNN系列网络模型，并且解释了目标检测领域的基本术语和主要的数据集。

## 目标检测概念综述

本节介绍了目标检测的基本概念以及发展现状。计**算机视觉(Computer VIsion)**主要指识别和理解图像的内容，包括分类、定位、目标检测、语义分割和实例分割，每一步都是在前一步的基础上进行的，如图 2‑1所示。**图像分类**是最常见的计算机视觉问题，对网络输入查看图像并对其中的对象进行分类。图像分类具有广泛的应用范围，从社交网络上的人脸检测到医学中的癌症检测，这些问题通常使用卷积神经网络建模进行分类就可以解决；**定位**进一步判断图像中的目标具体位置，只需要测出固定数量的预测框，即标记对象的类别，给出物体的确切位置围绕目标的位置绘制边框；**目标检测**需要对图像中的多个物体给出其的分类和定位。比如在自动驾驶汽车中，不仅需要检测汽车，还需要检测车架中的行人，摩托车，树木和其他物体。图像分类和定位是目标检测的基础；**语义分割**需要判断哪些像素属于哪个目标，将相同整体的所有像素预测同一类别，注意这是并不区分同种类别的不同物体的，比如第一个人，第二个人是不加区分地被分在一起的；**实例分割**需要区分出相同类别的所有实例，所有实例都需要被区分出来，即使是同一种类别的物体；除此之外，还有基于视频时序的**目标跟踪**等等，都是计算机视觉的问题。

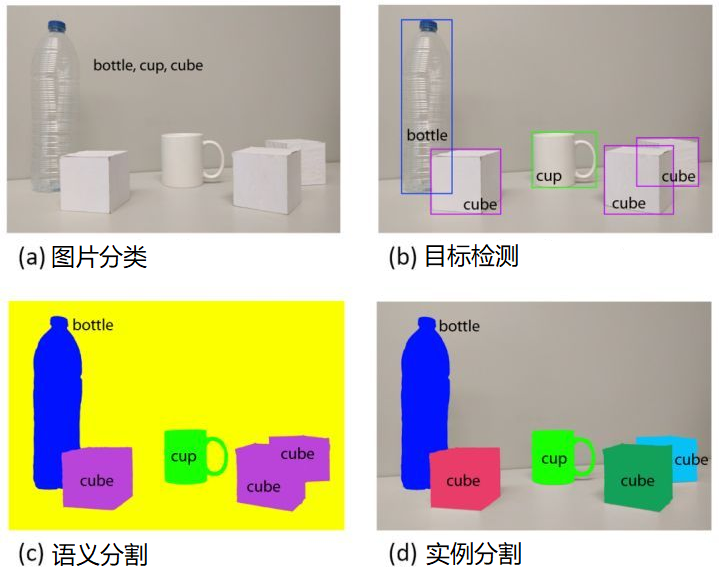


图 2‑1 计算机视觉主要任务

图像分类输入一张的是图片，输出的是物体的类别，用准确率来评判；而定位还需要输出检测到的目标位置，目前主流的目标检测算法主要是基于深度学习模型，其可以分成两大类，一步算法(one-stage)和两步算法(two-stage)。

1. 两步算法，顾名思义其将检测问题划分为两个阶段：定位和分类，首先产生候选区域，对候选区域分类（必要情况下位置还需要精修）。这类算法的典型代表是基于PRN的R-CNN系算法，如R-CNN，Fast R-CNN，Faster R-CNN等，由于注重的是区域选择，所以物体定位精确度高，但是速度比一步算法慢。
2. 一步算法，其不需要额外的区域选择阶段，同时回归网络产生物体的类别概率和位置坐标值。典型代表模型比如YOLO和SSD。由于没有额外的区域选择模块，一般速度很快，可以达到实时检测，但精确度不如两步算法。

目标检测模型的主要性能指标是**检测准确度**和**速度**，其中准确度包含定位准确度和分类准确度。目前目标检测领域有很多用于物体检测的预训练模型，比如本文使用的YOLO系列，R-CNN，Fast R-CNN，Mask R-CNN，SSD，Multibox等。

### 目标检测常用术语

目标检测模型评估条件分两个方面，一个是分类的准确度，一个是定位的准确度。单纯比较绝对坐标值很复杂，因此我们引入了**IOU**(**Interest of Unitied**)交并比来表示预测框和真实框的交叉部分比上预测框与真实框的并集部分（如图 2 2），值越大说明预测框与真实框的相交部分越多，也离真实框越近。当IOU超过某个阈值时，判定检测正确，否则认为检测错误。

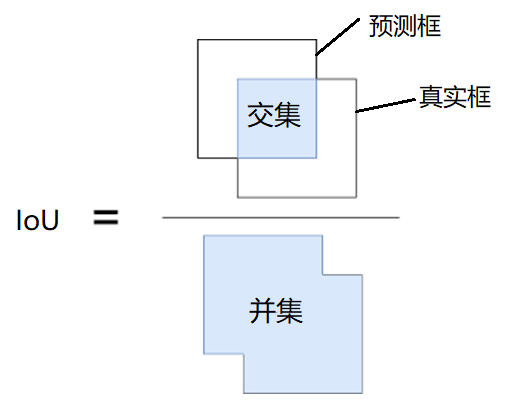


图 2‑2 IOU(Interest of Unitied，交并比)计算图例

由于是标注的数据，所以我们可以计算得某种类别的物体的测试精确度，假设有一张图片，对于给定的类，可以计算该图片的某种种类*C*的精确度如下：

其次，多种类目标检测中，有多种物体种类，若要计算某类别的精确度，则需要验证集中的每个图像计算，然后对其求平均，得到了单个类别的精确度：

最后为了用单个数字来衡量一个模型的表现，用所有类别的精确度求平均有：

这个值就是目标检测中衡量识别精度的指标，称为**mAP(mean average precision)**。多类别物体检测中，**召回率(recall)**指的是样本中真正正确的有多少被预测正确了，**准确率(percision)**预测为正确的样本中有多少是真正正确的。每一个类别都可以根据这两个值绘制一条曲线，*AP*(*Average Percision*)就是该曲线下的面积，*mAP*是多个类别*AP*的平均值。

除了定位精确度*mAP*，目标检测算法的另外一个重要性能指标是速度，只有速度快，才能实现实时检测，评估速度的常用指标是每秒帧率**FPS(Frame Per Second)**，即每秒内可以处理的图片数量。对比FPS需要在同一硬件上进行，因为CPU或者CPU计算能力对图片处理能力影响极大。另外也可以使用平均处理一张图片所需时间(ms)来评估检测速度。

### 目标检测常用数据集

表 2‑1列出了常见的大型图片数据集，包含各种各样的物体和类别，众多著名的算法及网络都是在这上面训练并测试的，本文使用的YOLO v2也是基于ImageNet和COCO训练及测试的。表 2‑2列出了常见的人脸数据集，本文也是借助了其中的FDDB与CelebA人脸数据集进行训练的。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 数据集 | 图片数 | 类别数 | 特征 | 最后更新 |
| ImageNet | 450K | 200 | 最大图像识别数据集之一 | 2015 |
| COCO | 120K | 80 | 图像识别和分割数据集 | 2014 |
| Pascal VOC | 12K | 20 | 物体检测和分类数据集 | 2012 |
| Oxford-IIIT Pet | 7K | 37 | 宠物图像数据分割数据集 | 2012 |
| KITTI Vision | 7K | 3 | 自动驾驶场景数据集 | 2014 |

表 2‑1 常见的大型图片数据集，包含各种物体种类

|  |  |  |  |
| --- | --- | --- | --- |
| 人脸数据集 | 图片数 | 特征 | 最后更新 |
| FDDB | 3K | 椭圆形标注人脸 | 2010 |
| AFLW | 25K | 图像识别和分割数据集 | 2017 |
| LFW | 13K | 物体检测和分类数据集 | 2012 |
| WIDER FACE | 3K | 宠物图像数据分割数据集 | 2017 |
| CelebA | 200K | 最大的人脸数据集之一 | 2015 |

表 2‑2 常见的人脸数据集

数据集有不同的格式

<http://www.360doc.com/content/17/0807/19/10408243_677391071.shtml>

## 常见的目标检测模型

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。

目标检测领域著名的R-CNN，Fast R-CNN, Faster R-CNN系列深度神经网络模型，其共同特点是two-stage的，需要先使用启发式方法（selective search）或者CNN网络（RPN）产生Region Proposal，然后再在Region Proposal上做分类与回归。

图片的空间位置信息的不变性，尽管卷积过程中图片大小减少，但是位置对应关系还是保存的

采用滑动窗口的目标检测算法思路非常简单，它将检测问题转化为了图像分类问题。其基本原理就是采用不同大小和比例（宽高比）的窗口在整张图片上以一定的步长进行滑动，然后对这些窗口对应的区域做图像分类，这样就可以实现对整张图片的检测了，如下图3所示，如DPM就是采用这种思路。但是这个方法有致命的缺点，就是你并不知道要检测的目标大小是什么规模，所以你要设置不同大小和比例的窗口去滑动，而且还要选取合适的步长。但是这样会产生很多的子区域，并且都要经过分类器去做预测，这需要很大的计算量，所以你的分类器不能太复杂，因为要保证速度。解决思路之一就是减少要分类的子区域，这就是R-CNN的一个改进策略，其采用了selective search方法来找到最有可能包含目标的子区域（Region Proposal），其实可以看成采用启发式方法过滤掉很多子区域，这会提升效率。

### Faster-RCNN系列模型结构概述

基于区域特征提取或者滑动窗口的模型都是基于**区域选择（Region Proposal）**的。OverFeat是2013年发布的纽约大学的一个早期目标检测模型，使用深度学习进行对象检测的第一项进展之一是他们提出了一种使用**卷积神经网络（CNN）**的**多尺度滑动窗口**算法。在OverFeat之后，加利福尼亚大学伯克利分校提出了R-CNN，其大体思想是使用区域选择（Region Proposal）方法（最著名的选择性搜索）选取最有可能的区域，使用CNN从每个地区提取特征，用SVM对每个区域进行分类。

### SSD系列模型结构概述

### YOLO系列模型结构概述

类one-stage算法，其仅仅使用一个CNN网络直接预测不同目标的类别与位置。YOLO算法很好的解决了这个问题，它不再是窗口滑动了，而是直接将原始图片分割成互不重合的小方块，然后通过卷积最后生产这样大小的特征图，基于上面的分析，可以认为特征图的每个元素也是对应原始图片的一个小方块，然后用每个元素来可以预测那些中心点在该小方格内的目标，这就是YOLO算法的朴素思想。

YOLO的CNN网络将输入的图片分割成 S\times S 网格，<https://zhuanlan.zhihu.com/p/32525231>

## 本章小结

总结来看，虽然YOLOv2做了很多改进，但是大部分都是借鉴其它论文的一些技巧，如Faster R-CNN的锚箱，YOLOv2采用锚箱和卷积做预测，这基本上与SSD模型（单尺度特征图的SSD）非常类似了，而且SSD也是借鉴了Faster R-CNN的RPN网络。从某种意义上来说，YOLOv2和SSD这两个one-stage模型与RPN网络本质上无异，只不过RPN不做类别的预测，只是简单地区分物体与背景。在two-stage方法中，RPN起到的作用是给出region proposals，其实就是作出粗糙的检测，所以另外增加了一个stage，即采用R-CNN网络来进一步提升检测的准确度（包括给出类别预测）。而对于one-stage方法，它们想要一步到位，直接采用“RPN”网络作出精确的预测，要因此要在网络设计上做很多的tricks。YOLOv2的一大创新是采用Multi-Scale Training策略，这样同一个模型其实就可以适应多种大小的图片了。

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YOLO的目标检测方法

本章是全文的重点章节，主要介绍了YOLO v2模型的概念及基本结构，将本文的目标检测模型过程进行了完整的解剖和描述。最后分析对比了该领域现存的几个著名模型与YOLO v2模型的差异。

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。

## YOLO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。

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Method** | **backbone** | **test size** | **VOC2007** | **VOC2010** | **VOC2012** | **ILSVRC 2013** | **Speed** |
| OverFeat |  |  |  |  |  | 24.3% |  |
| R-CNN | AlexNet |  | 58.5% | 53.7% | 53.3% | 31.4% |  |
| R-CNN | VGG16 |  | 66.0% |  |  |  |  |
| SPP\_net | ZF-5 |  | 54.2% |  |  | 31.84% |  |
| DeepID-Net |  |  | 64.1% |  |  | 50.3% |  |
| NoC | 73.3% |  | 68.8% |  |  |  |  |
| Fast-RCNN | VGG16 |  | 70.0% | 68.8% | 68.4% |  |  |
| MR-CNN | 78.2% |  | 73.9% |  |  |  |  |
| Faster-RCNN | VGG16 |  | 78.8% |  | 75.9% |  | 198ms |
| Faster-RCNN | ResNet101 |  | 85.6% |  | 83.8% |  |  |
| YOLO |  |  | 63.4% |  | 57.9% |  | 45 fps |
| YOLO VGG-16 |  |  | 66.4% |  |  |  | 21 fps |
| YOLOv2 |  | 448x448 | 78.6% |  | 73.4% |  | 40 fps |
| YOLOv3 |  |  |  |  |  |  |  |
| SSD | VGG16 | 300x300 | 77.2% |  | 75.8% |  | 46 fps |
| SSD | VGG16 | 512x512 | 79.8% |  | 78.5% |  | 19 fps |
| SSD | ResNet101 | 300x300 |  |  |  |  | 16 fps |
| SSD | ResNet101 | 512x512 |  |  |  |  | 8 fps |
| DSSD | ResNet101 | 300x300 |  |  |  |  | 8 fps |
| DSSD | ResNet101 | 500x500 |  |  |  |  | 6 fps |
| ION |  |  | 79.2% |  | 76.4% |  |  |
| CRAFT |  |  | 75.7% |  | 71.3% | 48.5% |  |
| OHEM |  |  | 78.9% |  | 76.3% |  |  |

## YOLO系列模型结构发展与变化

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。直接在输出层回归bounding box的位置和bounding box所属的类别，把整张图作为网络的输入，把 Object Detection 的问题转化成一个 Regression 问题。该模型有以下特点1. 速度快，能够达到实时的要求。在 Titan X 的 GPU 上 能够达到 45 帧每秒。 2. 使用全图作为 Context 信息，背景错误（把背景错认为物体）比较少。 3. 泛化能力强。

### YOLO v1模型结构概述

YOLO v1作为系列模型的初代，是YOLO系列模型的基础，其后所有系列模型的改进都是从YOLO v1最初的结构开始的。前文提到，YOLO模型和Faster-RNN等基于Region Proposal方法的模型不一样，YOLO结构完全是在一个单独的深度卷积神经网络中实现了目标检测，把检测框选择和类别分类放在了一个网络中。由于输入的是一整张图片，直接经过单个网络回归输出的是检测框，所以是端到端的神经网络模型。

YOLO v1模型首先将输入的图片重新调整为448×448像素大小，将图片分割为7×7网格，对每个网格来说，如果物体中心点落在该格内，则该格负责检测这个物体属于哪个种类，记为，表示确定网格包含物体的情况下属于的概率。

每个网格内含有两个限位框，**限位框**就是勾勒物体四周的目标结果框（图 3‑1所示），每个限位框用来表示位置与**置信度**，其中是限位框的中心坐标与网格左上角的位移，是限位框的宽与高。为了将这四个参数标准化使其取值落入[0, 1]区间内，最后还需要除以原始图片的长和宽；而指该限位框是否有物体落入，表示限定框与真实框位置的交并比(intersection over union)，即是限位框置信度。

当我们将限位框置信度与网格种类概率相乘时，有：

得到了限位框对于包含第类物体的置信度，即同时得到限位框（而不是网格）包含第类物体的概率和限位框有多准确地勾勒真实物体。

如果设定每个网格有2个限位框，这样得到了7×7×2=98个限位框，YOLO v1模型根据设定的阈值去除置信度低的限位框，最后再用非极大值抑制方法去除相近的限位框。由于一个网格负责一个物体，所以一个网格只属于一种分类，即每个限位框在测得坐标和长宽的同时，测得7×7=49个网格属于20种物体分类的概率。每个网格可以得到2个限位框×(4个坐标+1个限位框置信度)+20个分类概率的张量，因此7×7个格子共得到：7×7×(2×5+20)=7×7×30的张量。

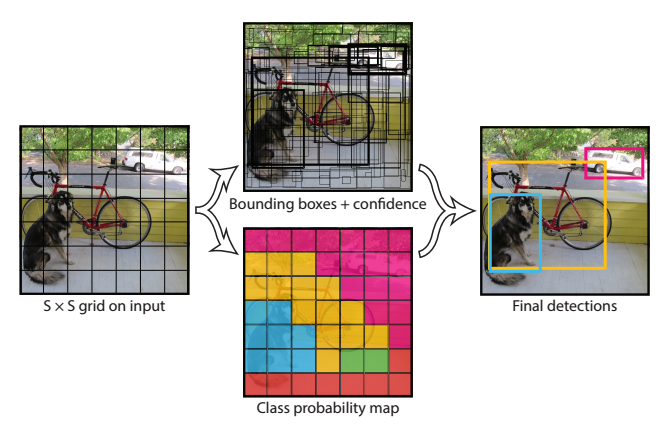


图 3‑2 图片被分割为7×7的网格，每个网格只检测一种类别（图中不同颜色被判定为不同类别），每个网格又只探测两个物体中心落在本格的限位框（图中黑线框）

和所有网络一样，YOLO v1通过卷积层来提取特征，通过全连接层来预测和坐标。YOLO v1模型网络受到了GoogLeNet模型的启发，采用了24个卷积层和2个全连接层，并且用1×1的卷积核代替了GoogLeNet的inception Module。将1×1卷积核跟在3×3卷积层后面。**1×1卷积核**有改变维度的作用：对于单通道输入张量来说，设置不同的padding和stride相当于对输入张量进行缩放操作；当遇到多通道时，由于一个卷积核对应输出结果张量的最后一维的维度数，如果维度数小于原输入通道则属于**降维**，反之升维，相当于跨通道的池化操作。相比传统的降维手段比如白化PCA，使用1×1卷积核降维的特点在于可以帮助减少模型参数（计算量），也可以对不同特征进行尺寸的归一化；也可以用于不同通道上特征的融合，同时保留了空间信息。

YOLO v1网络将目标探测问题看作一个限位框位置和置信度回归问题，模型采用均方误差作为损失函数。损失函数如图3-3所示：

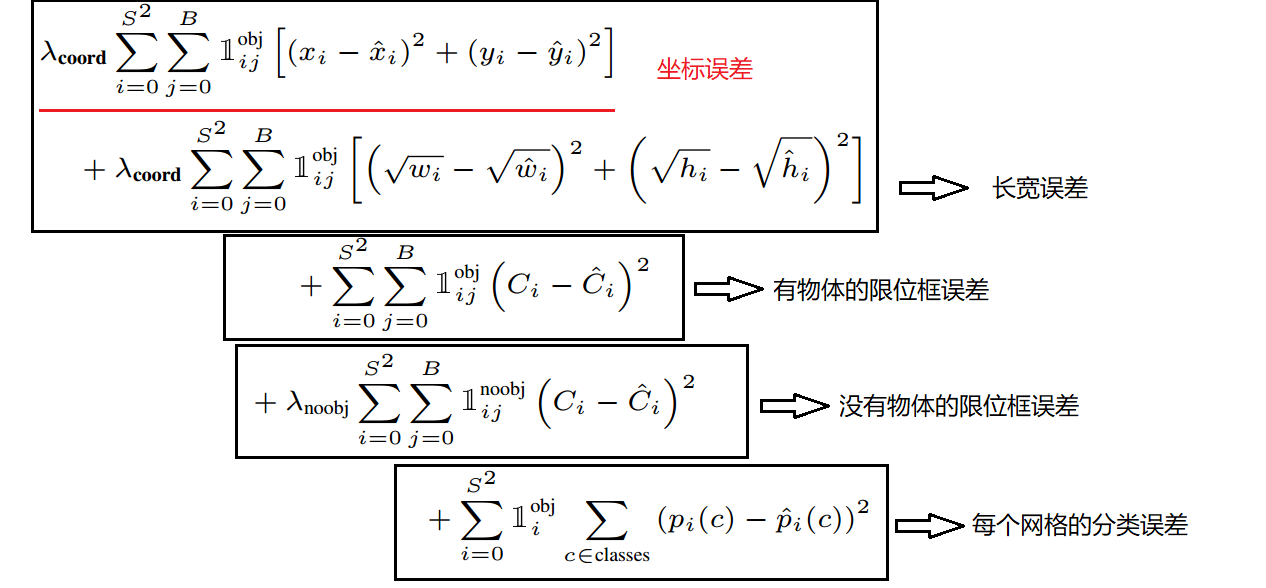


图 3‑3 YOLO v1模型的损失函数设置，采用均方差为基础，引入λ\_x设置不同的权重

采用均方误差会导致另一个问题：当网格不包含物体时置信度为0，导致梯度更新时传递0的网格很快覆盖掉其他有物体的网格的参数，使得网络不能收敛，所以网络通过引入参数的方式削弱了不包含物体的网格对损失函数的影响；另一方面，加重限位框坐标位置偏差对损失函数的影响，减少限位框分类偏差对损失函数的影响；在普通测试中发现，较小的限位框的坐标误差往往很敏感，限位框偏离一点就会与真实框在偏差很多（相同的偏差小限位框计算得到的损失函数更大），而大限位框的坐标误差并不敏感，为了均衡限位框大小对损失函数的影响，用长与宽的平方根来进行对比。

### YOLO v2模型结构概述

YOLO v1虽然检测速度很快，但是在检测精度上却不如基于区域选择的R-CNN系检测方法，在限位框定位方面精确度明显低于R-CNN系列。YOLO v1的另一个缺点在于对小物体的探测不足，由于每个网格只有2个限位框，因此当一个网格落入两个以上物体时，YOLO v1模型就无法探测出这些小物体并且可能会出现定位错误。因此在YOLO v2中使用了多种策略，在提升模型的定位的精准度的同时也要继续保持实时的检测速度。

首先，YOLO v2在模型中引入了可以加快模型收敛防止模型网络参数爆炸的著名的**批正则化层**(**BN层**, **Batch Normalization**)。神经网络学习权重参数的根本是为了学习输入数据的数据分布，要求模型需要具备良好的泛化能力，以适应不同的测试数据的分布。在最初的深度神经网络模型中，通常需要对**超参数**进行很多次的尝试分析择取最优，尤其是学习率和参数的初始化。在实际模型的迭代收敛中发现，当输入网络的训练数据每次被随机分为不同小批次(mini-batch)训练时，其分布都不尽相同并不是稳定的，而网络每一层的输入都是前一层的输出，那么在每次迭代梯度更新时模型都需要学习新的数据分布形式，在经过比如*sigmoid*这样的激励函数时会使深层反向传播的梯度变得趋于0近乎失效，*sigmoid*激励函数形式如下：

其导数为：

可以看到*sigmoid*激励函数只有在靠近*x*=0的部分导数才不为0（如图 3‑4），对于落在[-4,4]区间外的数据几乎变化很小导数趋近于0，那么根据链式传播法则，深层反向传播回来的残差回溯的层数越多，信息损失就越多，具体参见当误差结果从*h*层反向传播到第*k*层时，残差*L*对第*k*层输入求偏导过程，有：

其中*L*表示残差，表示第*k*层输入，表示第*k*层输出（），表示第i层权重，表示对第k层输出的加上激励函数处理。

考虑到除1以外的任何数的幂次最后会变得极小或极大（比如0.15=0.00001），由此每一层的权重和激励函数的导数和返回的残差在不断累乘最后也会变得极小或极大：激励函数的导数在0处最大只能取 ，所以在权重小于1的情况下连乘梯度更新结果会迅速趋近于0；而在大于1的情况下也有可能会出现梯度变化巨大的现象。事实上除开*sigmoid*激励函数，换作其他函数（单侧激励抑制的*Relu*，取值全为正的*tanh*，*sigmoid*是最简单的连续可导0-1跃迁函数），只要连续出现权重小于1的情况就有可能梯度更新结果会迅速趋近于0。总的来说，就是在深层累积学习到的残差在反向传播梯度更新时可能会失效或者变得无限大而使网络无法正常更新，即出现臭名昭著的**梯度弥散**或**梯度爆炸**，这将会大大降低网络的训练速度，并使得模型无法收敛。因此为保证模型在进行梯度更新前保证输入数据已经经过归一化，Google提出了BN层，经过BN层后输入数据被划分在均值为0，方差为1范围内，使得反向传播更新权重参数时，不同批次的数据对权重造成的改变不会互相影响，并且可以通过简单变换使数据理论上能够还原最初的形式，避免将已学习到的特征分布破坏。

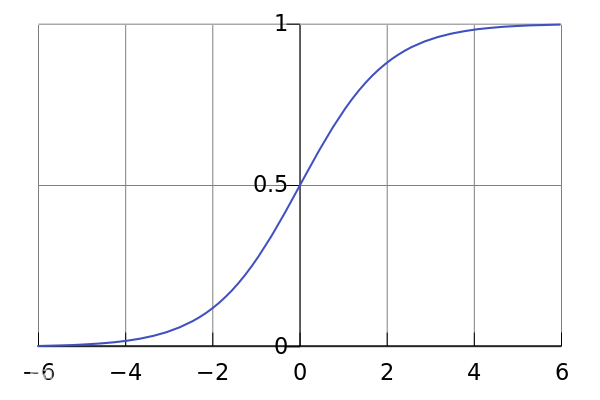


图 3‑5 sigmoid激励函数（也被称为Logistic逻辑回归函数），取值在[0,1]之间，在靠近0的部分增长最快，当绝对值＞5的时候几乎停止增长

YOLO v2使用了新的限位框生成预测策略。在YOLO v1模型中，是先直接使用全连接层来预测限位框的初始坐标，再在其后接上卷积层去提取特征，但是光凭全连接层来进行预测会导致丢失较多的空间信息，造成限位框定位不准，所以YOLO v2 去掉了全连接层，将整个网络变成一个全卷积网络，可以保存更多的空间信息，并且去掉了后面的一个池化层以确保输出的卷积核特征有更高的分辨率。Faster R-CNN模型是人工选取先验预测坐标作为限位框的初始坐标，而没有直接预测限位框的初始坐标，实际探测中比YOLO v1定位精确度更好。Faster R-CNN中的**区域建议网络**(**Region Proposal Network**)使用卷积层来预测**锚箱**(**Anchor Boxs**)的偏移和置信度。**锚箱**是使用9种的候选窗口：三种面积（1282，2562，5122）三种比例（1:1，2:1，1:2）来预测限位框相对先验框的绝对偏移位置。预测偏移值比预测坐标更容易，并使网络更易于学习，所以YOLO v2借鉴了锚箱的思想，在YOLO v1的基础上，改进为预测限位框中心点相对于对应网格左上角的相对偏移值。如图 3‑6所示，其中pw，ph代表了先验框的长度，tx，ty，tw，th表示限位框的位置和长宽。同时为了将限位框中心点约束在当前网格中，使用σ(sigmoid)激励函数处理偏移值，使用这样预测的偏移值可以落在[0,1]范围内。

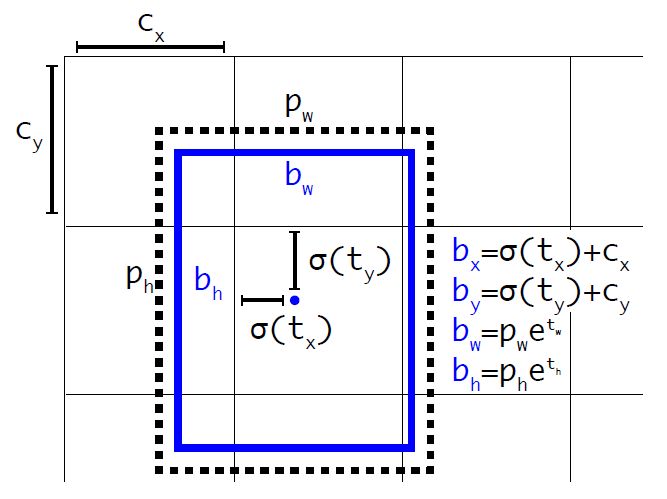


图 3‑7 YOLO v2先验框位置相对偏移预测

进一步改进了YOLO v1对物体尤其是小物体识别不全的问题。由于大物体通常占据输入图像的中间位置，YOLO v2可以只用中心的一个网格来预测这些大物体的位置。YOLO v2缩减模型结构让图片输入分辨率从之前的448×448改为416×416，这是为了让后面产生的卷积特征图的宽与高都为奇数，来产生一个中心网格（偶数会产生四个中心网格）。YOLO v2使用了因子为32的卷积层降采样，使得输入卷积网络的416×416图片最终得到13×13的卷积特征图（416/32=13），即一张图片最终被分为了13×13个网格，而YOLO v1是一开始固定大小的图片就被分割为7×7。对比在YOLO v1中是由每个网格本身来负责预测类别，网格对应的2个限位框负责预测坐标，在YOLO v2中限位框生成策略改为了每个网格对应k个（后文有实验表明最好是5个）限位框预测相对偏移位置和类别概率。根据锚箱策略，如果每个网格可以预测9个限位框，那么总共会预测13×13×9 = 1521个限位框，而之前的YOLO v1网络仅仅预测7×7×2 = 98个限位框，可以探测更多的物体。

在引入锚箱的时候YOLO v2遇到了两个问题，第一个问题在于：Faster R-CNN在初始化锚箱的宽与高的时候，是使用的手工精选的先验框。也就是说如果有更好的初始化锚箱策略，尽可能多地减少网络需要学习的迭代次数，为模型在初始化时选择更好的、更有代表性的先验框宽高，无疑会有更好的效果。因此YOLO v2使用了**K-means聚类方法**来判断需要多少个先验框，和其大概的初始化位置。第二个问题在于：传统的K-means聚类方法使用的是欧氏距离函数，这会使得较大的限位框会比较小的限位框产生更多的误差损失值（大数间的平方差远大于小数间的平方差）。因此YOLO v2将K-means聚类方法的距离函数改为和IOU（交并比）相关，和限位框的大小无关，IOU越大两个框相交得越多，通过与IOU来划分聚类，具体如下：

其中是第*i*个限位框与簇中心限位框的距离，而是第*i*个限位框与簇中心限位框的交并比。

最后，为了保持复杂度不太高的情况下先验框的平均IOU的比较高，要谨慎地选择k的取值，根据实验最后发现k=5时效果最好，即每个网格最好选择5个限位框，根据实验测试改进后的5个先验框策略与完全使用Faster R-CNN锚箱策略得到的平均IOU是相同的，如果再将先验框提升到9个，即每个网格9个先验框平均IOU会更好，但是网络会更大。并且统计发现聚类方法选择扁长的框较少，瘦高的框较多，即选择长宽2：1的较多。

13×13的卷积特征图对大尺寸物体的检测来说可能足够了，但是对小尺度的物体检测可能还需要加强一下。相对于Faster R-CNN和SSD直接在不同尺寸的卷积特征层中使用区域建议策略（Region Proposal）探测不同分辨率图片的不同大小的物体来确保小物体不会被遗漏，YOLO v2参考了Resnet网络的**Residual结构**（如图3‑8所示），选择增加一个**通过层**(**passthrough layer**)来得到更精细的特征层，加强对小物体的探测正确率，具体做法是将最后一层13×13×1024的卷积层前面的 26×26×512大小的卷积层进行隔行隔列地采样，通道增加4倍，变成了13×13×2048，两者进行叠加，使得邻接的特征被叠加到不同的通道中，经过扩张的特征图再进行下一步检测，加强了网络中图像细小的特征的信息，将高一级分辨率的特征连接到低分辨率，有利于检测小目标。

这里介绍一下ResNet中的Residual结构（如图 3‑9所示），越深层的卷积可表达的特征越丰富越抽象，但经过的网络层越深信息的损益也越多，反向传播的梯度在传播的过程中不断衰减使得错误率反而这个问题在层数很深的网络中很突出，即网络深度增加，网络效果反而变差甚至出现欠拟合。实验发现浅层的网络不容易出现该问题，因此如果能使深层的变化为浅层网络就可以解决。通过加上浅层的x来保证残差反向传播时尽可能地保存来自深层学习到的梯度信息，之前的层直接作为输入再次叠加到新层，再次记忆起网络浅层的输入流，对求导有，当时H，保证了梯度可以正常更新，减少了信息损益。

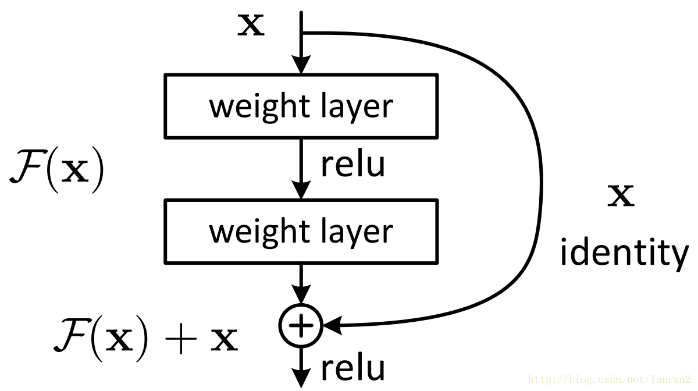


图 3‑10 ResNet网络中的Residual结构，通过加上浅层的x来保证残差反向传播时尽可能地保存来自深层学习到的梯度信息，经过的网络层越深信息的损益也越多。使用该结构可有效减少梯度衰减，配合达百层之多的深层神经网络梯度相关性得到了保持。

前文提到最初的YOLO接受输入的图片分辨率是448×448，加上锚箱后输入图片的分辨率改为416×416，为了使模型能够预测不同分辨率的输入图片，模型在训练的时候每隔10个批次会改变一下模型中输入图片的分辨率，由于还使用了因子为32的卷积层降采样，所以一般会随机选择是32的倍数的分辨率，最小的分辨率是320×320，最大的分辨率是608×608。因为没有固定输入尺寸而使用了不同输入分辨率的图片来训练，YOLO v2可以适应不同大小的图片，并且即使分辨率大也可以在精确度与速度上取得良好的结果（见表 3‑1）。

|  |  |  |
| --- | --- | --- |
| **目标检测网络模型** | **mAP** | **FPS** |
| Fast R-CNN | 70.0 | 0.5 |
| Faster R-CNN VGG-16 | 73.2 | 7 |
| Faster R-CNN ResNet | 76.4 | 5 |
| YOLO | 63.4 | 45 |
| SSD300 | 74.3 | 46 |
| SSD500 | 76.8 | 19 |
| YOLOv2 288 × 288 | 69.0 | 91 |
| YOLOv2 352 × 352 | 73.7 | 81 |
| YOLOv2 416 × 416 | 76.8 | 67 |
| YOLOv2 480 × 480 | 77.8 | 59 |
| YOLOv2 544 × 544 | 78.6 | 40 |

表 3‑2 YOLO v2不同分辨率下的速度与精确度，与其他目标检测网络模型的比较

YOLO v2将可测类别扩展到9000多种，并有效利用了现存的分类数据集作为训练集。目标检测领域的数据集的数据标注要远远比分类数据集的打单属性标签复杂，同理ImageNet分类数据集比VOC等目标检测数据集数量高出很多，并且有众多的维护者和使用者。由于YOLO中限位框的位置预测的是相对偏移，所以可以实现在分类和检测数据集上的联合训练。这种分类和检测的联合训练策略是指，对于检测数据集，可以用来学习预测物体的限位框、置信度以及为物体分类，而对于分类数据集可以仅用来学习分类。YOLOv2在训练网络时把目标检测数据集与分类数据集放在一起，遇到属于目标检测数据集的图片就基于的分类部分和检测部分的损失函数做反向传播；遇到属于分类数据集的图片就仅基于分类部分的损失函数做反向传播。

但是有一个问题需要解决：目标检测数据集的标签太抽象涵盖范围太广，例如“狗”；而分类数据集的标签更为细致，例如ImageNet就有一百多类狗的品种，大多数深度神经网络分类的方法是在最后的全连接层再接一个*softmax*来计算最终的概率分布，*softmax*计算的类别之间是互斥的、完全不相交的，但是融合的数据集之间的类别存在包含关系了，如何融合两类标签成了一个问题。YOLO v2提出了全新的**层级分类方法**(**Hierarchical classification**)，根据各个类别之间的从属关系建立一种树结构，称为**WordTree**（图 3‑11）。大多数分类数据集的分支关系转化为关系树后全部分类都只是互不相关的子节点。每个节点的子节点都属于同一子类，这时它们是互斥的，可以进行*softmax*计算分类。

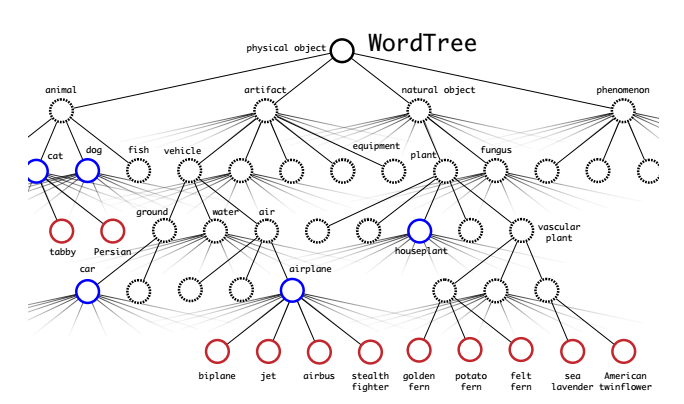


图 3‑12 WordTree的树形结构，同一节点的子节点是互不相关的，因此可以用softmax计算分类概率，分类数据集的分支关系转化为关系树后全部分类都是互不相关的子节点，被标记为节点会继承它所有父节点的属性

WordTree参考了WordNet，WordNet是基于ImageNet分类的**有向图**，对每一个种类，在WordNet上寻找从该种类所在节点到根节点的**最短路径**，将其加入层次树结构，循环这个过程，最终得到了 WordTree。对指定节点即分类种类的绝对概率的计算，有：

我们假定根节点，因此分类种类概率的计算只需要沿着根节点到该节点做连续乘积。YOLO v2用限位框位置和树结构中的概率和类别作为最终输出结果，在层级分类树结构中寻找概率之积最高的路径，当概率之积达阈值时停下，停止处的节点就表示结果类别。注意被标记为节点会继承它所有父节点的属性，当遇到新的或未知物体类别，仍然可以判断大的种类，而不能继续判断明确的是什么分类。使用树状图的另一个好处是可以融合不同数据集的标签，无论是分类数据集还是目标检测数据集。YOLO v2使用ImageNet的前9000类和COCO检测数据集以4：1的比例创造了一个联合数据集并打乱顺序。和之前的策略一样：

1. 当遇到检测数据集图片，对于其他损失部分如常操作，而对于分类损失部分，只在当前路径所有经过的节点分类上进行反向传播，不能传播到子节点，因为没有更精确的信息来了解是否属于子节点的分类情况。比如已知是“狗”，则“狗”之前经过的节点比如“哺乳动物”都会一起传播损失，但是子节点没有经过不能传播信息，比如“猎犬”，“小型犬”就不能传播因为不知道是属于哪一种分类。
2. 当遇到分类数据集图片，仅反向传播分类损失。假设限位框与真实框至少满足IOU >0.31，取该类别所有限位框中置信度最高的来进行分类预测，同样只在当前路径所有经过的节点分类上进行反向传播。

用这种联融合的方法，YOLO v2从检测数据集中学习物体位置，分类数据集中学习更广泛的、可扩展的物体分类。由于分类数据库数量巨大， YOLO v2只从大多数测试数据集中看到过分类数据而非检测数据。最终YOLO v2在ImageNet的评估数据集上测试获得的的精度为19.7mAP，在156个从未在任何目标检测数据集见过的物体检测数据类别上测试获得的精度为16.0mAP，高于**DPM**(**Discriminatively trained deformable part models**)，同时还可以以实时的速度检测9000多种其它物体类别。实验发现YOLO v2学习新动物表现很好，但是学习衣服和设备的表现不好，这是因为从COCO数据没有只有”人”(Person)这一个整体的类别，任何别的细致的标签数据，所以模型很难对“太阳镜”，“裤子”这些类别进行预测，如果要检测这一类别的物体，需要使用标注相应部位的数据集重新进行训练。

YOLO v2采用了全新的网络结构。大多数深度神经网络目标检测模型是基于VGG-16的，但VGG-16作为特征提取器计算复杂度太高，它的卷积层仅通过一次224×224图片就需要306.9亿次的浮点计算。YOLO模型是基于GoogLeNet的，GoogLeNet比VGG-16快，它的前向传递只需要85.2亿次浮点计算，虽然YOLO的普通模型在ImageNet获得了88.0%的准确度，逊于VGG-16的90.0%的准确度。所以在YOLO v2中采用了全新的**Darknet-19网络**，Darknet-19有19层卷积层，5层最大池化层，令人称赞的是Darknet-19只需要55.8亿次浮点计算，其在ImageNet上的top-1准确率为72.9%，top-5准确率为91.2%。Darknet-19吸收了很多其他网络比如VGG模型的结构的优点，比如使用了较多的3×3卷积核，每进行一次池化都把通道数翻倍，参考**NIN**(**Network in Network**)使用了**全局平局池化**(**Global Average Pooling**)操作来预测，新增了三个 3×3×2048卷积层，添加1×1卷积核插进在3×3卷积层之间来减少参数，在最后一层3×3×2048卷积层前新增一个通过层，同时使用BN批正则化层来加快收敛速度，提升模型鲁棒性和泛化能力。最后使用 1×1 卷积层输出预测结果，输出的通道数等于每个网格的锚箱限位框个数k×(2个相对偏移坐标+2个相对偏移长宽+置信度+分类概率向量) ，即：

YOLO v2模型成型过程首先是在ImageNet分类数据集上预训练Darknet-19，模型初始输入图片分辨率设定为 224×224，在预训练了160轮后，然后将网络的输入分辨率提高到 448×448 ，继续在ImageNet数据集上调优分类模型，训练10轮后在ImageNet上的top-1准确度已经达到76.5%，而top-5准确度是93.3%，并且可以达到实时检测。

### YOLO v3模型结构概述

## 移动端深度神经网络模型应用的优化方法

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

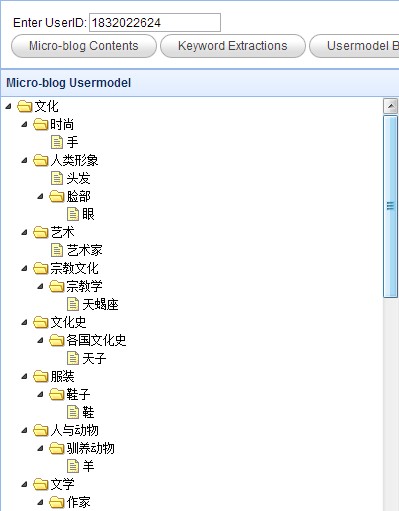
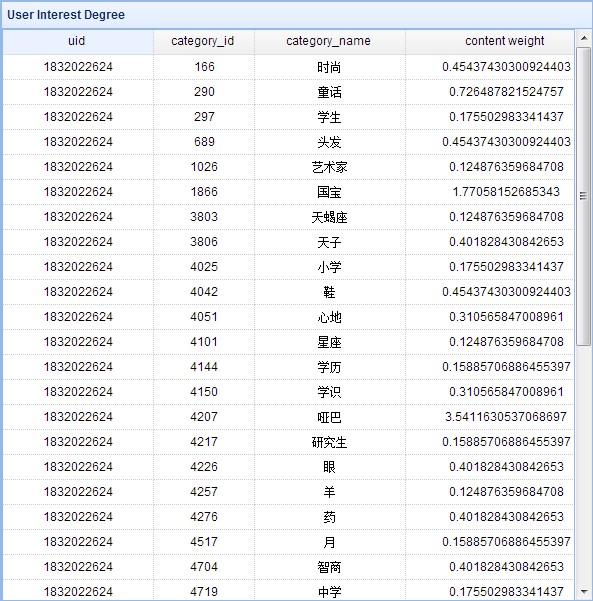


图3‑13用户模型中一个用户的兴趣树

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‑3‑14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。

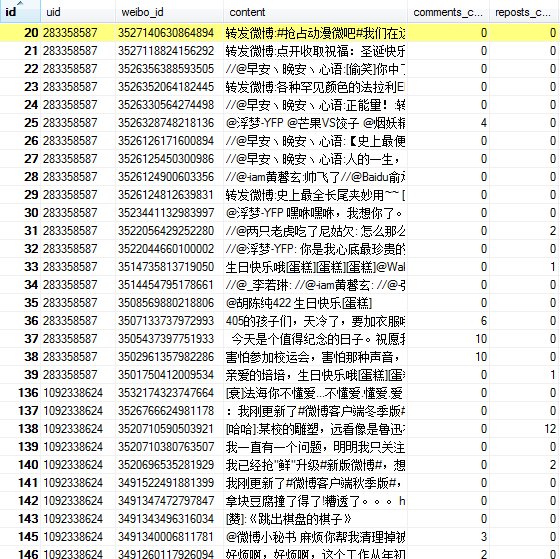


图4‑2微博内容数据表

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。

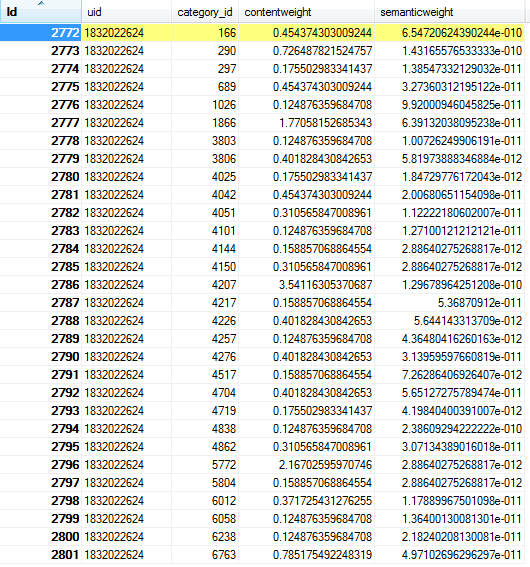


图4‑5用户模型数据表

## 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。<https://blog.csdn.net/gzs0927/article/details/78408467>

<https://zhuanlan.zhihu.com/p/35325884>

# 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>