**面向重复记录检测的数据清洗算法的研究**

**作者姓名 张 攀**

**学校导师姓名、职称 刘志镜 教授**

**企业导师姓名、职称 吴春苗 高工**

**申请学位类别 工程硕士**

**面向重复记录检测的数据清洗算法的研究**

**西安电子科技大学**

**硕士学位论文**

**作者姓名：**张 攀

**领　　域：**计算机技术

**学位类别：**工程硕士

**学校导师姓名、职称：** 刘志镜 教授

**企业导师姓名、职称：** 吴春苗 高工

**学　　院：**计算机学院

**提交日期：**2017年6月

**学　号　 1503121785**

**密　级　 公开**

**学校代码 10701**

**分类号 TP39**

Supervisor: Liu Zhijing

Supervisor: Wu ChunMiao

Title: Professor

Title: Senior Engineer

A thesis submitted to

XIDIAN UNIVERSITY

in partial fulfillment of the requirements

for the degree of Master

in Computer Technology

**Research of Data Cleansing Algorithm for Duplicate** **Elimination**

By

Zhang Pan

June 2017

**西安电子科技大学**

**学位论文独创性（或创新性）声明**

秉承学校严谨的学风和优良的科学道德，本人声明所呈交的论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果；也不包含为获得西安电子科技大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同事对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

学位论文若有不实之处，本人承担一切法律责任。

本人签名： 日 期：

**西安电子科技大学**

**关于论文使用授权的说明**

本人完全了解西安电子科技大学有关保留和使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权属于西安电子科技大学。学校有权保留送交论文的复印件，允许查阅、借阅论文；学校可以公布论文的全部或部分内容，允许采用影印、缩印或其它复制手段保存论文。同时本人保证，结合学位论文研究成果完成的论文、发明专利等成果，署名单位为西安电子科技大学。

保密的学位论文在 年解密后适用本授权书。

本人签名： 导师签名：

日 期： 日 期：

摘要

随着信息技术的发展与信息化建设的深入研究，激烈的市场竞争对于企业信息化程度的要求也越来越高。在形形色色的企业信息化系统进行不断地迭代和重构时，数据库中积累了大量的脏数据[1]，主要包括错误数据、相似重复数据和缺失数据三种类型。这些脏数据产生的原因多种多样，包括数据来源不同，存储于不同的操作系统以及硬件平台等[2]。其中多源数据的归并造成的数据重复则是最关键的热点问题[3]。

本文在对相似重复记录技术发展和研究现状进行简要介绍的基础上，首先详细地阐述了基于不同实现方式的重复检测技术，然后在借鉴前人工作成果的前提下，提出和设计了一种改进的IMPN算法。该算法主要有三个改进点：通过统计字段区分度一定程度上改善了传统的MPN[4,5]算法在选择排序关键字时过于依赖专家经验的缺点；通过动态调整滑动窗口的步长与大小以节约时间消耗；通过标记排序关键字为空的记录提高算法应对缺失字段的能力，增强了鲁棒性。

其次当数据量增大到一定量级的时候，算法的效率问题随之凸显。本文将Java中的多核并行框架Fork/Join应用到IMPN算法中，经实验对比可以验证其高效性。多核并行并没有改变算法是基于排序-归并[6,7]思想的事实，所以该解决方案依旧存在算法的检测精度对于排序关键字过于依赖的问题。针对大数据量的相似重复记录检测，本文提出一种改进的人工神经网络算法。首先通过已知小数据集作为训练数据对BP神经网络进行训练，然后利用遗传算法弥补其收敛速度慢和易陷入局部极小值的缺点，改进后的神经网络对于相似度向量输入可以迅速得到记录是否相似的判断。该算法的主要优势在于并不需要直接估算各属性的权重，而是通过神经网络学习属性之间的内在关系来完成记录是否匹配的判断，随着环境的变化具有较强的自适应能力。经对比试验证明，该算法拥有较好的检测效果。

**关键词**：数据清洗，相似重复记录检测，BP神经网络，遗传算法，多核并行计算

ABSTRACT

With the development of information technology and information construction, fierce market competition drives the enterprises to improve their information management. During the process of iteration and reconstruction of EIS, databases have accumulated a lot of “dirty data”, which consists of error data, duplicate data and missing data. There are varies of reasons of these data, such as the different data source, the different OS, the different hardware and so on. The key issue of these data cleansing problem is deduplication caused by the merge of multi-source data, which becomes the research focus nowadays.

In this paper we introduce the development and research status of the duplicatoin detection technology firstly, including their theory and implemention. Then a promoted algorithm called IMPN is present on basis of previos work. There are three improvement, decreasing the dependence of expertise and experience by the statistic of fields’ discrimination when choosing sort key, decreasing time consumption by dynamically adjusting the size and step length of sliding window, and promoting rubust by marking those records whose sort key is null..

The efficiency becomes a fierce problem as the data grows too large. Fork/Join, the multi-core parallel computing framework, is used to improve its performance. It is the fact that this solution is based on the Sort-Merge method, which depends much on the choice of sort key and perform not so good at time consuming, although with the Fork/Join framework. This paper provides a promoted artificial neural network algorithm for the deduplication of large dataset. Firstly, a small dataset is used to be the input of a BP network. After training process, the network will evolve as the chromosome, then this network could rapidly decide weather two records are the same entity with the similarity vector being the input. This solution’s advantage is that it gets the result through the inner relation of records rather than estimates the fields’ weight directly, which enhances its self-adaption ability as the environment changes. This algorithm reaches a higher accuracy as experiments prove.

**Keywords:** Data Cleansing, Duplicate Record Detection, BP Neural Network, Genetic Algorithm, Multi-core Parallel Computing

插图索引

[图2-1 Flume组成结构示意图 5](#_Toc482816758)

[图2-2 Flume数据流向示意图 5](#_Toc482816759)

[图2-3 Flume总体架构示意图 6](#_Toc482816760)

[图2-4 Flume抽象数据流向示意图 7](#_Toc482816761)

[图2-5 Kafka架构示意图 8](#_Toc482816762)

[图2-6 Kafka消费组示意图 9](#_Toc482816763)

[图2-7 Storm总体架构示意图 11](#_Toc482816764)

[图2-8 Storm工作时序图 12](#_Toc482816765)

[图2-9 HDFS整体架构示意图 13](#_Toc482816766)

[图2-10 MapReuce整体架构示意图 14](#_Toc482816767)

[图2-11 HDFS读取文件时序图 15](#_Toc482816768)

[图2-12 HDFS写入文件时序图 16](#_Toc482816769)

[图3-1 User-CF算法基本原理示意图 19](#_Toc482887542)

[图3-2 Item-CF算法基本原理示意图 21](#_Toc482887543)

[图3-3 Memory-base CF基本过程示意图 21](#_Toc482887544)

[图3-4 Model-Based CF基本过程示意图 22](#_Toc482887545)

[图3-5 SVDFeature模型原理示意图 25](#_Toc482887546)

[图4-1 系统总体需求示意图 29](#_Toc482887674)

[图4-2 日志采集系统需求示意图 30](#_Toc482887675)

[图4-3 推荐系统需求示意图 32](#_Toc482887676)

[图4-4 推荐方式组成示意图 33](#_Toc482887677)

[图4-5 个性化推荐算法流程图 35](#_Toc482887678)

[图4-6 UI需求示例图 36](#_Toc482887679)

[图5-1 实时推荐系统的总体结构示意图 39](#_Toc483579301)

[图5-2 流式计算流程图 40](#_Toc483579302)

[图5-3 实时推荐系统的数据流向示意图 41](#_Toc483579303)

[图5-4 日志采集系统框架示意图 43](#_Toc483579304)

[图5-5 日志采集数据序列图 46](#_Toc483579305)

[图5-6 数据接入模式架构示意图 47](#_Toc483579306)

[图5-7 Kafka生产者示意图 48](#_Toc483579307)

[图5-8 Flume的配置示例图 48](#_Toc483579308)

[图5-9 推荐算法计算过程流程图 51](#_Toc483579310)

[图5-10 Storm流式计算框架示意图 54](#_Toc483579311)

[图5-11 实时推荐实现流程图 57](#_Toc483579312)

[图5-12 小型电商APP界面展示图 58](#_Toc483579313)

[图5-13 推荐准确率柱状图 60](#_Toc479433389)

[图5-14 推荐召回率柱状图](#_Toc479433389) 60

表格索引

[表2-1 相似重复记录的典型例子 31](#_Toc477465047)

[表5-1 用户浏览信息表 42](#_Toc482800994)

[表5-2 用户检索信息表 42](#_Toc482800995)

[表5-3 用户收藏信息表 42](#_Toc482800996)

[表5-4 用户购买记录信息表 43](#_Toc482800997)

[表5-5 Flume配置方案表 44](#_Toc482800998)

[表5-6 数据源表 52](#_Toc482800999)

[表5-7 物品评分表 52](#_Toc482801000)

[表5-8 物品出现次数表 52](#_Toc482801001)

[表5-9 物品同现矩阵表 53](#_Toc482801002)

[表5-10 物品-用户评分矩阵表 53](#_Toc482801003)

[表5-11 推荐结果表 53](#_Toc482801004)

[表5-12 集群环境配置表 59](#_Toc482801004)

[表5-13 推荐计算时间表 61](#_Toc482801004)

[表5-13 不同算法计算时间表 61](#_Toc482801004)

符号对照表

exp 以e为底的指数函数

min 取最小值函数

∑ 求和

缩略语对照表

EIS Enterprise Information System 企业信息化系统

OS Operating System 操作系统

BP Back Propagation 反向传播

MPN Multi-Pass Sorted Neighborhood 多趟近邻排序算法

IMPN Improved Multi-pass Sorted Neighborhood 改进的多趟近邻排序算法

SNM Sorted Neighborhood Method 紧邻排序算法

ANN Artificial Neural Network 人工神经网络

目录

[摘要 I](#_Toc507624050)

[ABSTRACT III](#_Toc507624051)

[插图索引 V](#_Toc507624052)

[表格索引 VII](#_Toc507624053)

[符号对照表 IX](#_Toc507624054)

[缩略语对照表 XI](#_Toc507624055)

[第一章 绪论 1](#_Toc507624056)

[1.1 研究的背景和意义 1](#_Toc507624057)

[1.2 国内外研究现状 2](#_Toc507624058)

[1.3 论文研究的主要内容 3](#_Toc507624059)

[1.4 论文结构 4](#_Toc507624060)

[第二章 重复记录检测相关算法概述 5](#_Toc507624061)

[2.1 相似重复记录概述 5](#_Toc507624062)

[2.1.1 相似重复记录介绍 5](#_Toc507624063)

[2.1.2 相似重复记录清洗过程 6](#_Toc507624064)

[2.2 字段相似度匹配算法 8](#_Toc507624065)

[2.2.1 字段匹配算法 8](#_Toc507624066)

[2.2.2 编辑距离算法 10](#_Toc507624067)

[2.2.3 Smith-Waterman算法 11](#_Toc507624068)

[2.2.4 N-Gram算法 11](#_Toc507624069)

[2.2 相似重复记录检测算法 12](#_Toc507624070)

[2.2.1 近邻排序算法 12](#_Toc507624071)

[2.2.2 多趟近邻排序算法 14](#_Toc507624072)

[2.2.3 优先队列算法 15](#_Toc507624073)

[2.3 算法的衡量标准 15](#_Toc507624074)

[2.4 本章小结 16](#_Toc507624075)

[第三章 基于MPN改进的IMPN算法 19](#_Toc507624076)

[3.1 Memory-CF模型 19](#_Toc507624077)

[3.1.1 User-CF模型 19](#_Toc507624078)

[3.1.2 Item-CF模型 20](#_Toc507624079)

[3.2 Model-CF模型 22](#_Toc507624080)

[3.2.1 Logistic模型 22](#_Toc507624081)

[3.2.2 LFM(SVD)模型 23](#_Toc507624082)

[3.2.3 FM模型 25](#_Toc507624083)

[3.3 系统推荐算法的优化 25](#_Toc507624084)

[3.4 本章小结 27](#_Toc507624085)

[第四章 Fork/Join并行框架在算法中的应用 29](#_Toc507624086)

[4.1系统功能性需求 29](#_Toc507624087)

[4.1.1 日志收集系统 30](#_Toc507624088)

[4.1.2 推荐系统 32](#_Toc507624089)

[4.1.3 UI系统 36](#_Toc507624090)

[4.2本系统的重点需求 36](#_Toc507624091)

[4.2.1 实时性问题 36](#_Toc507624092)

[4.2.2 冷启动问题 37](#_Toc507624093)

[4.2.3 时效性和多样性问题 37](#_Toc507624094)

[4.3 本章总结 38](#_Toc507624095)

[第五章 改进的遗传神经网络算法 39](#_Toc507624096)

[5.1 整体框架 39](#_Toc507624097)

[5.2 日志采集模块 42](#_Toc507624098)

[5.2.1 日志数据类型 42](#_Toc507624099)

[5.2.2 日志采集系统框架 43](#_Toc507624100)

[5.2.3 日志采集系统连接器 44](#_Toc507624101)

[5.3 数据接入模块 47](#_Toc507624102)

[5.3.1 数据接入框架 47](#_Toc507624103)

[5.3.2 数据接入连接器 48](#_Toc507624104)

[5.4 离线计算平台模块 50](#_Toc507624105)

[5.4.1 基于物品的协同过滤算法原理 50](#_Toc507624106)

[5.4.2 算法实现过程 50](#_Toc507624107)

[5.5 流式计算平台模块 54](#_Toc507624108)

[5.5.1 Kafka与Storm的连接器 54](#_Toc507624109)

[5.5.2 实时推荐计算 55](#_Toc507624110)

[5.6 数据存储和展示模块 57](#_Toc507624111)

[5.6.1 数据存储和输出 57](#_Toc507624112)

[5.6.2 UI展示 58](#_Toc507624113)

[5.7 实验与测试 58](#_Toc507624114)

[5.7.1 评测标准 58](#_Toc507624115)

[5.7.2 实验设计 59](#_Toc507624116)

[5.7 本章小结 62](#_Toc507624117)

[第六章 总结与展望 63](#_Toc507624118)

[参考文献 65](#_Toc507624119)

[致谢 67](#_Toc507624120)

[作者简介 69](#_Toc507624121)

# 第一章 绪论

## 1.1 研究的背景和意义

信息化时代人类正在由工业化时代进入信息化时代，经济学家们普遍认为，进入21世纪后，信息将成为第一生产要素，同时将构成信息化社会的重要技术物质基础。为了在激烈的市场竞争中占据先机，各行业如保险、金融等纷纷加快了信息化的步伐。随着数据库技术的快速发展和广泛应用，形形色色的企业信息化系统应运而生。数据库的大小也与日逐增，人们已经得出如下结论，每过20个月世界上信息的数据量就会翻一番。

从规模庞大数据库中提取重要信息，从而对企业单位的发展提供参考，为决策者提供技术支持，是近年来数据挖掘的研究重点。然而由于不可避免的人为录入错误，或者是不同的数据表示方法，抑或是从不同的数据源合并数据甚至数据存储于不同的操作系统和物理设备，都不可避免地降低了系统的数据质量，从而产生各种“脏数据”。脏数据的类型主要包括重复数据、不完整数据、错误数据等。对这些数据若不进行清洗，则会影响信息化系统的正确运行，使得数据中提取的信息不再可靠，为企业决策支持和商务应用带来负面影响。因此，为了确保数据的准确性、一致性，数据清洗显得尤为重要。

最早的数据清洗过程需要大量的人为操作，所以当遇到较大规模的数据集，就会凸显出人为操作的低准确性和低效性。所以在当前数据规模急剧加大的情况下，只有借助信息技术，数据清洗才能实现其高效性。目前的信息化清洗过程中，仍不能完全离开专家的经验、人工的操作等行为，所以研究的一个重要方向就是尽可能减少人为的参与和影响。

相似重复的记录是数据库中降低数据质量最重要的一个原因，所以如何高效地检测和去除重复数据是数据清洗研究范畴的一个热点问题[8,9]。

同一个实体在数据库中不同的展现形式是相似重复记录的本质，它主要会引发以下的问题：

(1)资源浪费：重复记录会造成数据冗余，导致存储空间的极大浪费。

(2)破坏数据一致性：相似重复记录之间的关系可能是互为补充，也可能存在部分的冗余，甚至互相矛盾。当它们共同对应的现实中的实体发生变化，极有可能导致这些记录中只有某个或者某些记录发生改变，而其余无法同步更新。

以上，相似重复记录的检测与消除，保证了数据的一致性、减少资源的浪费，是数据清洗的重要环节。

## 1.2 国内外研究现状

早在上个世纪50年代，数据清洗已经开始了相关研究。将出自不同数据源的数据集进行整合自研究开始以来就被认为是一个困难而且极为重要的问题，开始主要是从数据连接[10]、数据实体识别[11]、对象识别等问题来研究，是商业保险、医疗、等领域中的研究重心之一。美国当时清除全美社会保险号数据集中的错误数据被视为数据清洗技术研究的开端[12]。

数据清洗的研究重心在以下几点：重复记录检测、异常数据检测、缺失数据的处理。数据仓库的出现以及数据挖掘相关技术的发展和应用，造成了多源数据进行合并容易出现大量重复数据的问题。因而相似重复记录的检测与清除成了数据清洗领域的研究重点。

在重复记录清洗方面，国外展开了大量的研究，主要的工作在于两个方面——属性匹配和重复检测。属性匹配问题的解决方法主要有Smith-Waterman算法、递归属性匹配算法、和R-S-W算法[13]。

相似重复记录检测大部分的算法采用排序然后归并的思想，即先将数据连接成一整个数据集，之后按照某种规则进行排序，将相似重复的记录排列在附近，最后通过某种相似判断方法检测出重复的记录，主要的差异在于排序方法和相似检测方法。最基本的算法是Jaro[14]提出的“排序&合并（Merge/Purge）”算法。这种算法存在明显的缺点，许多研究人员在此基础上提出了各种各样的改进思路和算法实现。

Monge等[15]将数据库中的一条记录视为一个字符串，在排序和比较的时候采用优先级队列的方法，检测相似重复时则使用了基于字符串的编辑距离。Hernandez等[5,4]提出了一种名为“多趟近邻排序”的算法，即MPN（Multi-Pass Sorted Neighborhood），该算法多次执行SNM（Sorted Neighborhood Method）同的排序关键字段以及较小的滑动窗口，最后判定记录相似时候使用的是一种规则定义的判等理论，它是使用C语言重写的OPS5[16]规则编程。Newcombe等[17]则采用应用相关的键值以将重复记录聚类到相近的位置。Qiu等[18]则是首先计算每条记录的N-gram统计值，然后根据这个N-gram值对数据集进行排序，最后再用用优先级队列的方式聚类检测重复记录。Gianni Costa等[19]采用文本聚类中的增量技术将新数据划分到最近的已知重复的聚类中，目的是为了解决大文本库中的相似检测问题。Alfredo Ferro等[20]使用了基于q-grams的相似度衡量函数，可以避免许多不必要的比较和判断，提高了时间效率。

国内的相关研究主要是对已知算法的改进和创新以实现更高的精度和效率。复旦大学周傲英等[21]等比较早开始数据清理的研究工作。邱越峰等[22]提出了基于N-Gram的相似记录检测算法。算法以一条数据的N-Gram值作为排序键，在对因为拼写造成的重复记录进行聚类时表现良好。!!!1韩京宇等[23]提出了一种基于q-gram空间层次聚类的方法。不同于传统的“排序&合并”的方法，该算法将数据映射成q-gram空间中的点，形成具有相似性层次子空间序列，采用层次聚类来实现相似重复记录的检测和清洗。方法不仅能避免应外排序引起的大量 I/O，而且克服排序不能保证将相似记录排在邻近位置的缺陷，还能通过层次逐步细化聚类来检测相似重复记录的方法提高检测精度。陈伟[24]提出一种按字段等级划分并将等级转变成权重的方法检测相似重复记录，给各个属性赋予合适的权重以提高检测精度，同时提出了长度过滤方法减少不必要的编辑距离计算进行相似重复记录检测优化。李星毅、庞雄文和周丽娟等[25,26,27]对大数据集相似记录检测问题。采用某种方法确定关键属性，依据关键属性进行数据分组，然后再划分后的组内进行相似重复记录检测的研究。这两种方法不同之处是确定关键属性的方法和组内聚类比较方法。

近几年，国内外的数据清洗市场的发展很快，其中包括商业上的数据清洗软件,也有各大学和研究机构开发的数据清洗软件[28,29]。

综上所述，国内外研究人员对数据清洗尤其对相似重复记录清洗的检测的研究已经得到取得了诸多进展，但仍旧或多或少存在适用面窄或者检测效率和精度不足等问题。

## 1.3 论文研究的主要内容

从国内外的研究现状可以看出，数据清洗领域己取得发展，促进了各行各业的发展。但数据清洗领域还存在一定问题，主要体现在：

(1)尽管检测重复记录受到很大关注，采取了许多措施，但检测效率与检测精度问题并不令人满意。大数据量的相似重复检测消耗大量时间，有待于更好的检测算法。

(2)大多数数据清理工具都是针对特定领域，其应用受到一定限制，未来特定领域的数据清洗仍是应用重点，但通用的清理方案应受到越来越多的关注。!!!1

(3)传统的相似重复记录检测大多基于“排序&归并”的思想，排序的效果以及最终归并的结果受排序关键字影响较大，尤其是当数据库排序关键字对应的字段为空或者是错误数据时，部分重复记录无法被正确的检测到，从而影响数据清洗的质量。

如何高效地检测相似重复记录，进而剔除数据库中的冗余数据，一直是数据清洗研究的重点问题。本文在分析了常用相似重复记录检测算法的基础上，针对传统的基本紧邻排序算法MPN在时间消耗和检测精度的不足，提出了改进的IMPN算法。IMPN算法的改进点主要在于：

(1)通过统计字段区分度一定程度上改善了传统的MPN算法在选择排序关键字时过于依赖专家经验的缺点。

(2)通过动态调整滑动窗口大小以节约时间并提高检测精度。

(3)通过标记排序关键字为空的记录提高算法应对缺失字段的能力，增强了鲁棒性。

为了节约IMPN算法的时间消耗，本文将多核并行框架Fork/Join应用到改进后的算法中，经过充分的对比实验可以验证其高效性，从而充分地利用了多核多线程计算机的优势。

随着人工神经网络研究的兴起，越来越多的跨学科研究正在如火如荼地展开。本文将训练后的反向传播神经网络应用于相似重复记录检测中，利用两条记录对应字段间的相似度，构建基于神经网络的检测模型，利用遗传算法对网络模型的权值进行优化，使用遗传神经网络组合多个字段上的相似度来检测相似重复记录。在多个测试数据集上的测试结果表明，该方法能够提高相似重复记录检测的准确率和检测精度。

## 1.4 论文结构

论文的结构如下所示：

第一章，介绍了的数据清洗研究的背景和意义，相似重复记录的国内外的研究与发展现状，简单描述了论文的主要研究目的以及研究内容，展示了论文的组织架构。

第二章，主要介绍了相似重复记录检测的相关算法。第一部分首先简单介绍了衡量字段相似度的相似度检测有关算法，分析了它们各自的有缺点以及适用条件。然后在第二部分首先介绍了最原始的直观方法，然后对最基本的紧邻排序算法SNM，介绍了算法的基本原理、设计思路以及大致实现和算法的优缺点。然后对在SNM基础上改进的MPN算法的设计原理、实现与优缺点分析进行了描述。除此之外还介绍了常用的算法思路比如优先队列算法等。

第三章，首先介绍了IMPN算法的提出背景，然后详细介绍了算法的设计思路和其改进点，并采用SNM算法和MPN算法作为对照，进行了对比实验以验证算法的效率和检测精度等优势。

第四章，首先介绍了神经网络的基础知识以及神经网络在相似重复记录检测中的应用，接下来介绍了多种对于BP神经网络的改进思路，重点介绍了遗传算法对其的改进，并说明了如何使用改进的BP神经网络进行相似重复记录检测。通过使用不同数据量的测试数据集进行改进证明该算法的优势，并分析了它的缺点。

第五章，介绍了多核并行的基础理论，将Fork/Join并行框架应用到IMPN的设计与实现思路，并与IMPN做对比实验，验证多核并行实现对于算法在时间效率上的提高。

第六章，总结了本文的内容包括IMPN算法与遗传神经网络进行重复检测研究中遇到的问题，以及算法本身的不足之处，并对未来的研究进行展望。

# 第二章 重复记录检测相关算法概述

## 2.1 相似重复记录概述

### 2.1.1 相似重复记录介绍

!!!2由于数据输入错误、不标准的缩写词，或其它原因，数据库中可能包含关于现实世界同一实体的重复记录。虽然关系数据库系统不允许含有重复主键值的记录输入，但是，由于数据输入错误，不管主键的值是否被这些错误影响，关系数据库不能再保证不存在重复的记录。因此，在数据清理中，相似重复记录的检测与清除是一个重要问题。

为了便于给相似重复记录下定义，下面首先看一个关于的重复记录实例，如表2.1所示。

表2.1 学生信息表中的重复记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Num | Name | Sex | Brithday | Department |
| 20110001 | John Smith | Male | 1985/02/01 | College of Computer Science,Guilin University of Electronic Science and Technology |
| 20110002 | Harry Potter | Female | 1988/08/12 | College of Computer Science,Guilin University of Electronic Science and Technology |
| 20110003 | J. Smith | M | 1985/2/1 | College of Computer Science,Guilin University of Electronic Science and Technology |
| 20110004 | Harry Potter | Female | 1988/08/12 | College of Computer Science ,Guilin University of Electronic Science and Technology |
| 20110005 | Mr.John S | Male | 1985-02-01 | Corllge of Computer Science ,Guilin University of Electronic Science and Technology |

表2.1关于学生信息的五条记录。表2.1中学号为2011002和2011004这两条记录所有属性值完全相同，因此，这两条记录可以认为是完全重复记录。而表中学号为2011001、2011003和2011004这三条记录看起来不相同，但实际上只是姓名属性值“John Smith”、“J. Smith”、“Mr.John S”书写形式不同但都是同一个人；性别属性的值“Male”、“M”采用了全称和缩写两种表示男性，出生日期属性的值“1985/02/01”、“1985/2/1”、“1985-02-01”采用不同格式书写表达式同一个日期；学院属性值中出现了最常见的拼写错误把“College”拼写成“Corllge”；可以认为三条记录描述的是同一个学生。因此，这里的三条记录可以认为是相似重复记录。根据以上分析，数据源中的重复记录可分成完全重复记录和相似重复记录，分别描述如下：

(1)完全重复记录[30]：指在数据表中除了主键外，其它各属性完全相同的记录，或者没有主键，所有属性完全相同的记录。

(2)相似重复记录[30]：指客观上表示现实世界同一实体的，但是由于在格式、拼

写上有些差异而导致数据库系统不能正确识别的记录。

### 2.1.2 相似重复记录清洗过程

数据清洗的目的就是首先通过某种科学的方法检测出相似重复记录，然后根据合适的规则和策略自动或手动清除这些相似重复记录。分析相关研究资料可知，完全重复记录检测大多采用“排序&合并”的方法：首先将数据库中的记录进行排序，重复的记录就会重新排序在相邻位置，通过对相邻记录比较，就可以检测出完全重复记录。这一方法被扩展得到相似重复记录的清洗方法的过程：数据调入→记录排序→记录相似检测→相似重复记录合并/清除。相似重复记录清洗过程如图2.1所示。



图 2.1

如图2.1所示，相似重复记录清洗过程描述如下：

(1)数据调入。通过数据库连接接口把含有相似重复记录的数据源中的数据调到系统中；

(2)相似重复记录检测。如果数据规模较大应从算法库中调用排序算法，执行记录之间的排序；在记录已排序的基础上，记录相似检测模块从算法库中调用相似检测算法，作邻近范围内记录间的相似检测，从而计算出记录间的相似度，并根据预定义的重复识别规则，来判定是否为相似重复记录。为了能检测到更多的重复记录，一次排序不够，要采用多轮排序比较，每次排序采用不同的键，然后把检测到的所有重复记录聚类到一起，从而完成重复记录的检测；若数据规模较小可以直接进行记录相似检测；

(3)数据合并/清除。对所检测出的每一组相似重复记录根据预定义的合并/清除规则，完成相似重复记录的合并处理。

由图2.1可知，记录排序和相似重复检测是相似重复记录清洗的两个核心步骤，本文中重点讨论相似重复记录检测的技术。

## 2.2 字段相似度匹配算法

为了从数据集中检测并清除重复记录，首要的问题是如何判断两条记录是否是重复记录。一条记录由不同的字段组成，字段之间的相似度考察是衡量两条完整记录相似与否的主要方法。目前相似重复记录的识别算法(也称为匹配算法)主要有：字段匹配算法（Field Matching Algorithm）、编辑距离法（Edit Distance Algorithm）、聚类算法（Clustering Algorithm）、N-Grams算法、对XML数据的匹配算法（Matching Algorithm for XML）等。

### 2.2.1 字段匹配算法

字段匹配是用来确定两个字段值是否表示同一个语义实体的句法上的可替换者, 是记录匹配的基础。基本的字段匹配算法和递归的字段匹配算法是两种最基础的字段匹配算法。

基本的字段匹配算法是基于传统的相似度匹配算法发展而来的[31]，能够有效的识别出包含前缀在内的相似重复记录，下面给出几个关于基本字段匹配算法的定义：

(1)元字符串：元字符串是指利用字符序列中的标点符号，空格以及其他分割符号，将字符序列分割成一段段以单词词组为中心的字符串[32]。

(2)字符序列的匹配度计算方法如下：

 公式2-1

基本的字段匹配算法的算法思想是将一条记录看作是一个字符序列，然后按照某些定界符将这个字符序列分割成一段段的字符串，也就是元字符串，然后通过基本的字段匹配算法公式，计算其匹配度。在这个过程中，元字符串相互匹配的标准主要有两种：一种是两者之间完全相同，每一个字符及其在其中的位置都相同；另一种情况则较为特殊，其中一条字符串的前几位和另一条字符串完全相同。只有在这两种情况下，才能计入匹配的数目中。两条记录间相似的元字符个数比上两条记录总的字符序列，得到的结果便是两条记录的基本字段匹配算法的相似度。

表2.2相似重复记录的典型例子

|  |  |  |  |
| --- | --- | --- | --- |
| Num | Name | Address | Tel |
| A | Join Kow | Comput Sci & Eng Dept, University of California, San Diego | 888-1234 |
| B | K. Join | Department of Computer Science, Univ. Calif, San Diego | 8881234 |

如表2.1中两个地址字段，A=“Department of Computer Science, Univ. CaliL, san Diego”，B=“Comput. sci. &Eng. Dept. university of California, san Diego”将这两条记录按照标点符号分割成小的字符串，其中，由于英文的阅读习惯，英文中的of在字段中作为标点符号使用，因此，经过分割操作，A和B分别变成如下两种情况：

A={Department，Computer，Science，Univ，Calif，san，Diego}

B={Comput，sci，Eng，Dept，university，California，san，Diego}

对他们两者进行分析可知，两条字段共有6条相似字符序列，分别为A的Computer，Science，Univ，Calif，san，Diego和B字段的Comput，sci，university，California，san，Diego。其中sci是Science的前缀。则两条记录的相似度为6/(|A|+|B|)/2)=0.8。

基本字段匹配算法有着很大的缺陷，根据它的算法思想，当一条字符串是另一条字符串的前缀缩写时，则将两者看作相似，但是，名词缩写并不仅仅局限于前缀或后缀缩写，更多的情况是类似于英文姓名缩写，取有代表性、并且和已存在的名词不重复的字符来代替整条字符串，当出现这种情况的时候，该算法并不能很好的识别出来。如上述A字符串中的Department的缩写是B中的Dept，相似重复记录的数据清洗系统中，这两个字符串被认为是相似数据，然而在基本的字段匹配算法却不能处理。但是，基本字段匹配算法却很直观，采用先分割后计算的方式，首先截取原记录，然后根据两者之间原子匹配数来计算相似度，其效率及准确度较其他匹配算法，有着很大的优势，为了保留这种优势，Minton等人对基本字段匹配算法进行了改进，提出了递归字段匹配算法。

递归字段匹配算法是一种较为准确的算法，能够有效的识别出含有缩写等情形的相似重复数据。它是基本字段匹配算法的升级，通过给一个字符串中的分割符以及标点设置优先级，并按照优先级的不同，从高到低，来对字符串进行分割，并且运用嵌套功能，不断重复这个过程，直到分割出来的子串不能被分割为止。利用这种递归的方式，将一条字符串分割成许多的原子字符串，然后再采用递归的方式，从上到下，不断匹配每一对的原子字符串，从而得到他们的相似度[33]。每两条字符串之间的相似度都是他们的子串的相似度的平均值，递归字段匹配算法的计算公式如公式2-2：

 公式2-2

其中，Ai、Bj分别是分割AB后得到的子串，、分别表示字符串A，B中的子串的个数。该算法适用于名词缩写和顺序颠倒等环境，能够有效的识别出含有这些情况的相似重复数据，例如上一个例子中的两条字符串匹配问题中，Department与B中原子字符串的相似度的最大值是与B中的Dept相比的4/7=相同字符个数/两者总共个数，computer最大值是6/7，science的最大值是3/5，univ的最大值是4/7,calif的最大值是2/3，san的最大值是1，Diego的最大值是1，则字符串A和字符串B相匹配，两者的相似度为0.75238,可以认为两者为相似重复记录。但是，递归字符匹配算法的缺点也很明显，与基本字段匹配算法相比，大量的使用嵌套递归功能，使得该算法的时间复杂度变高，是子串个数的平方；当字符串较长是，匹配每一个原子字符串都需要遍历另一个字符串，效率低下，而且对于包含大量错误字符的字符串记录，该算法并不能很好的识别。

### 2.2.2 编辑距离算法

Levenshtein等人在1965年提出一种基于编辑距离的相似重复记录匹配算法，又被称为L-距离算法[34]。编辑距离算法是Levenshtein等人所提出的一种用于发现相似重复数据的一种先进算法，他的主要思想是通过比较两条记录A和B，对他们进行分析，找到两者在字符上的不同，从而计算从记录A变化到记录B所需要经过的插入、删除、替换等操作的次数，该次数被称为编辑距离。所需要的操作次数越少，就代表这两条记录之间的编辑距离越近，两者之间的相似度就越大。因此该算法也被叫做L-距离算法。自从L-距离算法在1965年被提出以后，由于他的高效和高准，就得到了普及。例如两条记录children和child，如图2.2所示：



图2.2

记录children转换到记录child需要删除后面的三个字母r、e和n，共需要三个删除操作，则这两条记录的编辑距离为3。计算记录间的编辑距离算法在计算不同字符串的相似度上有着广泛的应用，它能够有效的识别出例如拼写错误、名词缩写等问题所产生的相似重复记录。由于编辑距离算法的高效性和广泛性，国内许多专家学者在它的基础上开发出更多的算法：刘旭辉等人在2008年在编辑距离算法的基础上，提出了一种基于NFA的编辑距离方法[35]，它将需要进行相似度计算的两条记录看作是一个二叉排序树，采用某种遍历方法，得到这个二叉数的有序序列，按照这个有序序列查找树进行索引，能够很大的提高识别效率和准确率；赵作鹏等人为了解决编辑距离算法无法解决单词顺颠倒等问题，在其基础上提出了一种改进的非相邻位置的编辑距离算法，能够有效的检测出单词替换和顺序颠倒等字符串之间的相似度[36,37]。

### 2.2.3 Smith-Waterman算法

S-W算法最开始被用于生物领域，用于匹配蛋白质和DNA序列，后来便逐渐被应用在相似重复记录的数据清洗中。S-W算法是一种动态编程技术，通过罚值和间隙计算不同记录之间的相似度。其算法的基本思想主要通过一个矩阵E来计算相似度，矩阵E是由a，b，c三个参数来确定的，其中，参数a为给定匹配字母表β的|β|\*|β|矩阵，字母表中第i个字符和第j个字符的匹配度放置在由字母表β组成的矩阵a的(i,j)位置上；b则为间隙开始的罚值；c为间隙间隙继续的罚值。矩阵E的组成元素(i,j),与字符串X和字符串Y的子串相关，他们的前i个和前j个字符组成他们的两个子串，这两个子串之间的最佳匹配值作为矩阵E的每一个元素，其计算公式为：

 公式2-3

当的值高于标准时，则认为X和Y这两者有着相同或相似的属性，互为相似重复数据。S-W算法可以有效解决字段匹配算法所包含的不足，对于包含不正确值的相似重复记录，能够非常准确的将他们辨别出来。但是对于名词缩写和前后顺序颠倒情况的相似重复记录，该算法未能取得较好的结果[38]。

### 2.2.4 N-Gram算法

基于N-Gram聚类算法的基本思想是：给每个记录赋一个N-Gram值，以该值为键来对记录聚类[39]。该算法能适应常见的拼写错误，从而较好地聚类相似重复记录。文献[22]提出了N-Gram标记算法和N-Gram聚类算法，用基于域的重复矩阵代替全局的重复矩阵，将相似记录聚到一个cluster中，并对每个cluster中的记录做改进的Pair-wise比较，从而提高了检测精度。文献[53]提出了一种基于N-Gram层次空间的聚类算法DGHS，定义记录r1、r2的N-Gram相似性，如下所示：

 公式2-4

其中表示记录r的所有属性组成的多重集合。通过将记录字符串根据依次加长的N-Gram映射到不同的子空间，采用层次聚类进行归并，从而实现相似重复记录的检测。

表2.3给出了几种N-Gram算法及其与传统排序-合并算法中常用的Merge/Purge算法[40]比较的情况。

表2.3几种N-Gram算法的比较

|  |  |  |  |
| --- | --- | --- | --- |
| 算法 | 时间复杂度 | 优点 | 缺点 |
| 基本的N-Gram算法 | O(n) | 能适应常见的拼写、交换、替换错误,是单词位置无关的 | 对插入、删除错误敏感，识别效果差 |
| 改进的N-Gram算法 | O(n\*c)c为常数，时间复杂度略高于基本的N-Gram算法 | 能自动校正单词的插入、删除错误，提高检测精度 | 时间开销大，对错误分布均匀的数据精度改进效果不明显 |
| 基于N-Gram层次空间的聚类算法 | O(h\*k)，k为簇的个数，h为聚类的趟数 | 能自动校正单词的插入、删除错误，提高检测精度避免外排序操作引起的大量I/O，对大数据量的处理具有良好的伸缩性 | 算法细节还需改进 |
| Merge/Perge算法 | O(nlgn) | 解决了如何检测数据库中完全重复记录的问题 | 对字符位置敏感，随着数据量增大算法精度会下降 |

## 相似重复记录检测算法

相似重复记录清除是在重复记录识别后进行的数据的合并或删除操作[41]，是影响数据质量的关键步骤。最简单直白的方法是对每条记录进行互相之间的比较，该方法的识别精度非常高，但是在数据量较大的情况下，其处理时间会让用户难以忍受[42]。目前的清除算法主要是采用排序-合并的思想，先将数据库中的记录排序，后通过比较临近记录是否相似来检测记录是否重复。常用的算法有近邻排序法（Sorted-Nighborhood Method，SNM）、多趟近邻排序法（Multi-Pass Sorted-Neighborhood，MPN）、优先队列算法（Priority Queue Strategy，PQS）、Delphi算法（Delphi Algorithm）等。

### 2.2.1 近邻排序算法

目前解决海量数据的记录相似匹配检测中，采用的较多的是近邻排序算法（Sorted-Neighborhood Method，SNM算法）。算法的基本思想是：将数据集R中的记录按指定的关键字（key）排序后，然后在排序后的数据集上移动一个固定大小的窗口，只检测窗口内的记录，判定它们是否匹配。由此来减少记录的比次数。SNM算法步骤为：

(1)生成关键字段：抽取表中的相关属性，生成关键字段；

(2)排序：按照生成的关键字排序对数据集中的记录排序；

(3)记录匹配：在排序后的数据集上滑动一个固定大小的窗口，数据集中的每条记录看做新记录仅与窗口内的记录进行比较，直到数据集的最后一条记录。

如果窗口的大小是包含N个记录，当窗口移动时，原来窗口中的第一条记录移出窗口，新进入窗口的记录与窗口中的旧的N-1记录相比较，判定是否匹配，如图2.3所示。



图2.3

算法执行时，首先要将数据级中的记录进行归一化处理，然后再进行记录的比较检测，下面给出算法实现，见算法2.1。

算法2.1最近邻算法SNM

输入：DB={t1,t2,…,tn} //数据集DB

A //表示成员之间距离的邻接矩阵

输出：K //簇集

K1={t1};

K={K1};

K=1;

For i=2 to n do

find the tm in some cluster Km in K such that dis(ti, tm) is the smallest;

IF dis(ti,tm)≤t Then //t 为阀值，由经验值确定

Km=Km∪ti

Else

k=k+1

Kk={ti}

SNM算法采用滑动窗口的方法，每次可以只比较窗口中的N条记录，提高了匹配效率；采用滑动窗口也极大的提高了比较速度，只需要进行N×n次比较，显然，N与n相比是小的多。但是SNM方法存在这样的两个缺陷[43,44]：

（1）对排序关键字的依赖性太大。SNM方法检测重复记录的精度很大程度上依赖所创建的排序的关键字，如果选取的关键字不当，邻近的位置不是相似重复记录，而使本来是重复的记录物理位置相距很远。

（2）滑动窗口的大小N的选择很难控制。当N值较大时，需要比较的次数较多，这样就增加了时间复杂度，N若较小的话，许多相似的记录可能就被漏配。当所有记录中各个重复记录聚类数差别较大时，则N的选取无论多大都不适合，除非N等于n。

### 2.2.2 多趟近邻排序算法

针对SNM算法存在的缺陷，Hernandez[4]等人提出了多趟近邻排序算法（Multi-Pass Sorted Neighbothood，MPN），该算法的基本思想是独立地执行多趟SNM算法，每趟创建不同的排序关键字和使用相对较小的滑动窗口。然后采用基于规则的知识库来生成一个等价原理，作为合并记录的判定标准，将每趟扫描识别出的重复记录合并为一组，在合并时假定记录的重复是具有传递性的，即计算其传递闭包(transitive closure)。传递闭包是指若记录Rl与R2互为重复记录，R2与R3互为重复记录，则Rl与R3互为重复记录。通过将每趟扫描识别出的重复记录计算传递闭包的方法，可以得到较完全的重复记录集合，能部分解决漏匹配问题。

但是MPN算法使用传递闭包容易引起误识别，针对这一缺点，文献[45]做了两个方面的改进：采用窗口大小可在Min和Max之间变化的可变窗口，通过相似度与阈值的比较及时调整窗口值，减少遗漏；不采用传递闭包，而引入有效权值，以消除字段缺失造成的负面影响。对于进行比较的两条记录A、B，假定参与比较的字段有n个，记录A、B的相似度计算方法如下所示：

 公式2-5

其中，。只有当两条记录在第i个属性上对应的值都不为空时，才进行字段比较，此时Valid[i]=1，对应的权值Wi为有效权值；否则Valid[i]=0。将总相似度与阈值比较从而判断重复记录，提高了识别的准确性，减少了误识别。其准确率几乎不受数据量的影响，一直保持在99%以上，取得了较好的识别效果。但参数的确定依赖领域知识，缺少特定的标准。

### 2.2.3 优先队列算法

优先队列法是由Monge等提出的[46]领域无关的处理相似重复记录的方法。用一个重复记录的优先队列来代替固定大小的窗口，优先队列的每一项代表的是一个重复记录聚类，而不是一条记录。算法顺序扫描记录，如果当前记录Rj本来就是优先队列对应簇中的成员，那么扫描下一条记录；如果不是，与优先队列包含的项比较，如在优先队列中有重复记录，则将这条记录合并入匹配记录所在的簇中，包含有这个记录的集合进入优先队列并有最高的优先级，如果扫描整个优先队列后发现，Rj不属于任何一个簇，则将Rj本身所在的聚类加入到优先队列中，并使其具有最高优先级，成为该聚类的第一个代表记录。优先队列越大，算法的检测精度越大，但是运行时间也越长。采用优先队列策略识别重复记录的精度很大程度上依赖于排序所选择的关键字。为了解决一个关键字不足以将所有重复记录聚集在一起的问题，可以每趟使用不同的关键字独立执行多趟优先队列算法，最后合并每趟扫描的结果。在合并时假定记录的重复具有传递性。图5描述了一个两趟式的优先队列扫描法[47]。

图2.4优先队列扫描算法

基于优先队列的算法以重复记录聚类为元素的优先权队列结合特征记录，通过适当的设置阈值，可以大大减少不必要的记录比较次数。而且该算法几乎不受数据规模的影响，能很好的适应数据规模的变化，但是检测结果的精度与这阈值的设定有很大的关系。而且排序算法的缺点在优先队列这里依然存在。

## 人工神经网络理论基础

【人工神经网络的理论基础+引入BP神经网络的原理】

人工神经网络简单模拟大脑处理信息的机制，它是由许多互相连接并传递信息的神经元组成的非线性处理系统，每个组成单元的结构功能并不复杂，整体却能以任意精度逼近线性或者非线性函数。如图4.1和图4.2所示，人工神经网络中的节点是对神经元的一种模仿：



图4.1 生物神经元示意图（引用自维基百科）



图4.2 人工神经网络节点

其中，~是输入向量的不同分量，~是神经元各个突触的权值，b表示偏置，是加权求和操作，则是激活函数。则对于输入向量经过此神经元时，经过加权求以及激活函数得到的输出为：。许多的类似于这样的神经元则组成了人工神经网络。

## 2.4 算法的衡量标准

衡量重复记录检测算法效率的标准，应该是算法是否能把数据集中存在的所有重复记录都检测出来。常用的标准主要有召回率，误识别率一和精确度。下面分别来介绍这几个度量的标准[48]。

(1)召回率（Recall）

也被称为百分比采样数。它定义为被重复记录检测算法正确识别出的重复记录占数据集实际包含的重复记录的百分比。其计算公式为：

召回率 = ( 正确识别的重复记录数 / 实际包含的重复记录数 ) \* 100%

假设我们有7条记录，A1，A2，A3，B1，B2，B3，C1，其中{ A1，A2，A3}与{ B1，B2，B3}分别是记录A和B的重复记录。通过一个数据清洗的过程识别出{A1，A2，C1}和{B1，B2}是重复记录，则其Recall=4/6\*100%=66.7%。

(2)误识别率（False-Positives）

有时被称为false merge，它的定义是被重复记录检测算法错误地识别为重复记录的数目占被算法识别为重复记录总数的百分比。误识别率越低表明算法结果的置信度就越高。其计算公式为：

误识别率 = (被错误地识别为重复记录的数目 / 被识别为重复记录的总数)\*100%

在上面的例子中，检测算法错误地把C1识别为一条重复记录，则其误识别率为1/5\*100% = 20%。

(3)精确度（Precision）

精确度是指识别出的重复记录表示的是否都是同一个实体，即是否存在误识别的情况。其计算公式为

精确度 = 1 –误识别率

一般而言，当召回率从0增加到100%时，精确率从100%减少到0。!!!2

## 2.5 本章小结

本章首先介绍了相似重复记录产生的原因和大部分去重算法的主要清洗过程，然后介绍了四种不同的相似度匹配算法：字段匹配算法、编辑距离算法、SW算法以及N-Gram算法。

其次本章重点介绍了几种不同的相似重复记录检测算法。从算法的设计原理，实现步骤，主要优缺点等方向对SNM算法、MPN算法、优先队列算法进行了介绍。

然后本章对于人工神经网络的理论基础以及BP算法的原理与设计思路做出了介绍。

最后本章介绍了衡量算法检测效率的几个重要指标。

# 第三章 改进的IMPN算法

多趟近邻排序算法（Multi-Pass Sorted Neighbothood，MPN）在传统的近邻排序算法（Sorted Neighbothood Method,，SNM）的基础上，通过计算每趟近邻排序算法重复检测结果的传递闭包，实现了更好的检测效果。它能够以更小的滑动窗口进行重复检测，并且可以检测到一些人工都难以发现的重复记录。

但是MPN并没有克服SNM算法对于排序关键字的选择标准过于依赖领域专家的经验，并且当记录的排序关键字字段为空时，检测效果会大打折扣。所以本文提出了基于统计规律的提取键值的IMPN算法，通过统计不同字段对于数据集记录的区分度，以及对空字段情况下的标记处理，可以一定程度上克服MPN对于专家知识的依赖，并提高了算法的鲁棒性。MPN算法在归并过程中采用的滑动窗口方法是继承自传统的SNM算法的，即固定的滑动窗口加固定的滑动速度。这种方式不够灵活，如果当前窗口内含有较少相似重复记录时则会出现较多的冗余比较，IMPN算法对于MPN算法的第三个改进点就在于采用了可伸缩大小的自适应滑动窗口，这种方式能够减少一些不必要的判等比较，从而节约时间，减少了算法时间复杂度。

## 3.1 基于字段区分度提取键值的方法

### 3.1.1 传统的提取键值的方法

基于排序/合并思想的重复记录检测算法，在排序之前需要提取和生成记录的排序键值，然后再对记录按照键值的顺序进行排序。完全理想状态下相似重复记录的键值也相等，所以排序过后相似重复记录则汇聚到邻近的位置，进而容易检测出相似重复记录对。

键值的定义是，从记录中提取出来的不同属性组成的序列或者属性的字符串子集[5]。键值的选择十分关键，因为只有选择恰当的排序键值才能够使得数据集在经过排序之后，相似重复记录聚集在排序后的数据集合中相邻的位置，这样在之后的滑动窗口中才能覆盖到相似重复的记录，合适的排序键值还可以减小之后排序使用的滑动窗口的大小。另外，不同的记录经过排序后应当距离较远，这样可以避免无意义的判等比较。综合来看，排序键值的选择不仅影响了算法的整体重复检测效果，还影响了算法的时间运行效率。

表3.1 四条相似重复记录的例子

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Record | First Name | Last Name | Address | ID |
| R1 | Jack | Stolfo | 123 First Street | 12345678 |
| R2 | Jack | Stolfo | 123 First Street | 12345673 |
| R3 | Jack | Stolpho | 123 First Street | 12345678 |
| R4 | Jacon | Stiles | 123 Forest Street | 12345432 |

举例来说，表3.1是四条相似重复记录R1，R2，R3，R4。它们由四个字段组成，分别是First Name、Last Name、Address和ID。可以看出R2和R1是完全相同的两条记录，因为姓名和地址完全一致并且ID可能由于印刷或者人为录入的错误只相差了一位；R3和R1对应的现实实体也是一致的，因为两者只有Last Name字段不一致，而且两者从发音上几乎一致；然而R4和R1则是对应着现实世界的两个不同的实体，因为Last Name字段完全不一致，而且两者的地址字段Address也相差甚远。所以可以得出R1=R2 =R3R4。

在下表3.1中，假设【方案一】的键值设计者选择的排序键值由以下几个部分组成：

（1）Last Name的所有部分

（2）Address的数字部分加字母部分的前三个辅音字母

（3）ID的前三个连续数字

那么它们对应的排序键值分别如表3.2所示：

表3.2 相似重复记录及其生成的键值

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Record | First Name | Last Name | Address | ID | Key |
| R1 | Jack | Stolfo | 123 First Street | 12345678 | STOLFO123FRT123 |
| R2 | Jack | Stolfo | 123 First Street | 12345673 | STOLFO123FRT123 |
| R3 | Jack | Stolpho | 123 First Street | 12345678 | STOLPHO123FRT123 |
| R4 | Jacon | Stiles | 123 Forest Street | 12345432 | STILES123FRT123 |

则可以看出该方案下，R1，R2，对应的键值相同均为“STOLFO123FRT123”，R1和R4对应的键值不同，经过排序后R1和R2聚集在邻近位置，R1和R4则不会处于邻近的位置，只看这两条记录是不能发现问题的。R1和R3原本是相同的记录，但是由于发音的问题导致在改方案情况下两者的排序关键字并不相同，所以按照【方案一】提取出来的排序关键字进行排序，很可能会导致R1=R4无法被正确的检测出来。

接下来再看另外一种方案。假设【方案二】的键值设计者选择的排序键值由以下几个部分组成：

（1）Last Name的前三个辅音字母

（2）First Name的前三个连续字母

（3）Address的数字部分加字母部分的前三个辅音字母

（4）ID的前三个连续数字

那么它们对应的排序键值分别如表3.3所示：

表3.3 相似重复记录及其生成的键值

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Record | First Name | Last Name | Address | ID | Key |
| R1 | Jack | Stolfo | 123 First Street | 12345678 | STLJAC123FRT213 |
| R2 | Jack | Stolfo | 123 First Street | 12345673 | STLJAC123FRT123 |
| R3 | Jack | Stolpho | 123 First Street | 12345678 | STLJAC123FRT123 |
| R4 | Jacon | Stiles | 123 Forest Street | 12345432 | STLJAC123FRT123 |

则可以看出该方案下这四条记录的排序键值均为“STLJAC123FRT213”。然而从表格中可以看出： R1和R2的排序键值的相同是合理的；R3和R1对应的现实实体也是一致的，选择辅音字母可以一定程度上客服发音上的错误，R1和R3的排序键值的相同也是合理的；然而R4和R1则是对应着现实世界的两个不同的实体，但是恰巧“Stolfo”和“Stiles”连续的三个辅音字母都是“STL”，“Forest”和“First”的前三个连续的辅音字母都是“FRT”，同理关键字段中对应的ID也是巧合，这种情况下可以看出【方案二】会导致R4和与其本身并不相同三条记录R1、R2、R3处于近邻的位置，从而会导致额外的判等计算，甚至如果判等方法不合适的话则可能错误地将R4与R1、R2、R3识别为重复记录，从而影响算法的准确性。

接下来再看另外一种方案。假设【方案三】的键值设计者选择的排序键值由以下几个部分组成：

（1）Last Name的前三个辅音字母

（2）First Name的前三个辅音字母

（3）Address的数字部分加字母部分的前三个辅音字母

（4）ID的前三个连续数字

那么它们对应的排序键值分别如表3.4所示：

表3.4 相似重复记录及其生成的键值

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Record | First Name | Last Name | Address | ID | Key |
| R1 | Jack | Stolfo | 123 First Street | 12345678 | STLJCK123FRT213 |
| R2 | Jack | Stolfo | 123 First Street | 12345673 | STLJCK123FRT123 |
| R3 | Jack | Stolpho | 123 First Street | 12345678 | STLJCK123FRT123 |
| R4 | Jacon | Stiles | 123 Forest Street | 12345432 | STLJCN123FRT123 |

则可以看出该方案下, R1、R2、R3的排序键值相同，它们对应的现实实体也是一致的。R4和R1则对应不同的排序关键字。这种情况下可以看出【方案三】是最理想的一种键值选择。

综合以上的三种情况不难发现：对于特定的数据集，生成排序键值的方式优劣取决于该数据所在领域的专家经验。比较合适的排序键值才能达到较为理想的检测效果和检测效率，这就需要大量的人工操作，工作量大且影响了算法的普适性。本文3.1.2提出了一种较少专家知识依赖的方法。

### 3.1.2 字段区分度方法

衡量排序键值优劣的一个原则即为不同的记录对应的键值不同，相同的记录对应的键值理应相似或一致。对于相似重复的记录，无论排序键值以何种方式选取，理论上生成的key值是相同或者相似的，所以衡量排序键值最关键的标准是在区分不同记录时的表现，即对应现实世界中不同实体的两条记录在合适的排序键值方案下生成的key值应当不一致。所以在选择排序键值时，应当选择有足够区分度的字段来提取键值。

如何衡量字段的区分不同记录的能力，这里提出“字段区分度”的概念。若数据库中一共有N条记录，每条记录都由M个字段组成，即，对于第i个字段fieldi，它的区分度如下公式3-1：

 公式3-1

其中，代表数据集中在fieldi字段一共有种取值，也就是将数据集按照fieldi进行聚类一共有簇。的取值介于0到1之间，值越高对应字段对于整体数据集的区分能力越大。提取排序键值之前首先计算所有字段的“字段区分度”值然后对其排序，排序键值每次选择值最大的一个字段的部分或者整体来作为键值的一部分。这里的“字段区分度”偏向于是一个相对的概念，因为相似重复记录在每个字段的内容理论上是相等的，即便是实际情况中由于印刷错误、格式不一致、人工录入等导致部分相似重复记录在某个字段表现不一致，这也是极少见的情况。因而无论数据集中的重复数据占到多大或者多小的比例，拥有较大区分度的字段的“字段区分度”值也较大。

在没有人工参与或者较少参与选取排序键值的情况下，利用“字段区分度”去选择排序键值的生成方式，有利于在排序后将对应不同实体的记录区分开来，将相似重复记录聚集到近邻位置，继而提高了算法的普适性。

## 3.2 自适应大小的滑动窗口检测方法

### 3.2.1 传统的滑动窗口检测方法

MPN算法的滑动窗口方法是继承自传统的SNM算法的。首先给定一个固定大小为w的滑动窗口如下图3.1所示：



图3.1 滑动窗口扫描过程

在排序之前首先要将数据源合并成一个线性序列，然后将这个滑动窗口从第一条记录开始，每次向下滑动一条记录同时原来w窗口的第一个记录被移出，直到最后一条记录进入窗口内。滑动过程中重复执行如下操作：刚进入的一条记录分别于之前的w-1条记录进行比较，判断是否是相似重复记录。需要注意的是，初始化w大小的滑动窗口之后，需要将最开始的这w条记录互相之间重复检测，因为后面的滑动过程中忽略掉了这前w条记录。

这种传统的固定滑动窗口存在一些问题，可以看出来滑动归并过程的时间复杂度为O(wN)，所以如果窗口的长度过大则会增加算法的的复杂度，长度如果太小又会降低检测的精度。

如下图3.2所示：



图3.2 不同滑动窗口大小时的归并过程

图中的record0\_origin是原始记录，record0\_dup0、record0\_dup1、record0\_dup2是其相似重复记录，而record1\_origin和record2\_origin是另外的两个不重复记录。若滑动窗口w的大小为2（这里为了举例说明而采用了一个比较极端的例子），按照传统的滑动窗口归并方法，无法检测到(4)与(1)、(2)是重复记录；若滑动窗口的大小为w2=5，虽然能够实现最好的检测效果，但是也最耗时间。可以看出(1)、(2)这段数据中重复率最高所以这时的滑动窗口应当较大，而当窗口移动到(3)、(4)、(5)的时候，记录互相不重复，所以这里窗口应当较小，以节约时间。所以总的来看MPN采用的这种滑动窗口方式还有很大的优化空间，最佳的方案应当是根据数据内容动态变化的滑动窗口。

### 3.2.2 自适应大小的滑动窗口检测方法

滑动窗口的大小可以根据当前滑动窗口的数据重复情况而做出动态地调整，当窗口w内的数据重复度比较高时，证明当前窗口整处于重复记录比较集中的位置附近，而且重复记录的数量可能更多，所以为了实现更精准的检测，窗口应当增大尺寸以包含位置比较远的相似重复；反之，若滑动窗口内的数据集重复度比较低，即相似重复数据较少时，说明当前位置附近的数据之间可能互相不重复，所以如果窗口较大的话就会造成这些互不重复的数据需要数次与新进入的数据进行判重检测之后才能退出滑动窗口，所以这时应当减小滑动窗口的大小。

现有的自适应窗口算法[51]大多是统计出当前滑动窗口内的所有的重复记录数，依据重复数据的个数占滑动窗口的大小来决定当前窗口的尺寸变化，此类方式存在很大的弊端：原本MPN的归并过程的时间复杂度只有O(wN)，其中w是滑动窗口的大小，N是待检测的相似重复记录集合中记录的条数，但是这种方法每次要对窗口内的记录进行一对一的重复检测，使得时间复杂度升高到O(N2)。

考虑到MPN原本在滑动窗口内的比较是将新进入的一条记录与窗口内仍保留的w-1条记录进行比较，当重复度较高的时候，下一次比较应当保证的是新进入的记录应该能够尽可能的照顾到即将要离开的记录，才能保证检测效果。换个角度说，在当前滑动窗口内，即将离开滑动窗口的记录若和新进入的元素是相似重复记录，这时应当扩大窗口尺寸。设w窗口内的重复记录在窗口内的位置依次为0、1、2...w-1，其中w-1是刚滑入的数据，0号位置的记录是即将被滑出的记录，则越是靠近0号位置的数据对于滑动窗口的尺寸影响越大。根据以上条件本文提出公式如下3-2：

 公式3-2

其中，滑动窗口的最大值为，最小值为，即将滑出的记录在数据集中的索引位置为，当前滑动窗口的大小为，代表数据集中索引为的记录是否与位置的记录互为重复，若是则=1，否则=0。可以看出，若内的记录都是重复记录，则下一个窗口的大小去最大值，相反若互不相似，则取最小值；并且距离越远位置的记录对下一个滑动窗口大小的影响越大（当其与位置的记录互为重复记录时）。

如下图3.3所示：

图3.3 可伸缩的滑动窗口

其中相同颜色的记录互为相似重复记录，归并过程从左向右进行。最大窗口为5，最小窗口为3，第①步窗口大小为4，到了第②步，由公式计算得到窗口变为5，因为窗口内的记录和R0\_2全部重复，第③步窗口大小又变成最小3，因为窗口内的数据和R1互不重复，④⑤⑥的变化同理。图上的例子很简明生动地表现了自适应滑动窗口的变化过程，展现了IMPN算法相对于传统MPN算法更加灵活高效的处理滑动窗口的方式。

## 3.3 基于标记的处理不完整排序键值方法

### 3.3.1 MPN的排序方法存在的问题

由于待清洗的数据集本身的数据质量并不高，所以记录中可能存在字段为空或者字段不完整的情况，如表格3.5所示：

表3.5 存在缺失数据及不完整数据的两条记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Record | First Name | Last Name | Address | ID |
| R1 | Jack | tolfo | 123 First Street | 12345678 |
| R2 | Jack |  | 123 First Street | 12345673 |
| R3 | Jack | Stolpho | 123 First Street | 12345678 |
| R4 | Jacon | Stiles | 123 Forest Street | 12345432 |

记录R1的Last Name字段原本应该是“Stolfo”，现在缺失了一部分成为“Sto”，在生成排序键值时若采用3.1.1中的方案三，即提取Last Name的前三个辅音字母是则只能得到“s”和“t”两个字母，则排序键值位数少了一位，所以提出的键值为“TLFJCK123FRT213”。记录R2的Last Name字段则直接完全缺失，所以提取的键值为“JCK123FRT213”，在排序的过程中，本来属于相似重复记录的R1和R2则由于开头字母的差异而被分到距离比较远的位置，从而可能无法被检测到。这个例子展示了不完整数据和缺失数据对MPN算法检测效果的影响，本文3.2.2提出了针对不完整排序键值的改进方法。

### 3.3.2 改进的基于标记的方法

缺失数据的处理是数据清洗的另一个分支研究领域，面对缺失值常见的做法主要有三种[49,50]：

（1）以同一指标的计算结果（包括均值、中位数、众数等）填充缺失值

（2）以业务知识或者经验填充缺失值

（3）从本数据集或者其他来源的数据集推测出来

其中第（1）种做法填充结果不够精细甚至过于粗糙，对检测结果可能造成负面影响；第二种做法填充结果可能较为准确但是需要人为的参与，工作量较大；第三种做法对数据集的数据质量要求较高并且能达到的效果下限很低。考虑到出现排序键值为空的记录只是影响到排序后记录的位置，所以本文提出了基于标记的处理方法：先将遇到的带有不完整键值的记录的ID进行标注，然后在其余完整键值的记录归并结束之后，对这些标注过的记录分别进行检测，将其一一聚类到已知的重复数据簇中。

本文这种基于标记的做法能弥补MPN算法面对缺失排序键值情况检测效果较差的不足，但是对于对含有缺失字段的记录占数据集比例较低的数据集合进行操作时不会耗费太多时间，但是当数据集中的大部分记录都含有关键字段的缺失时，该方法会导致时间复杂度升高，这是一个不足之处。

## 3.4 IMPN算法设计

### 3.4.1 算法流程设计

结合3.1~3.3的内容可以看出，IMPN算法的设计思想在于以下三点：

（1）引入“字段区分度”概念减弱算法对专家知识的依赖

（2）采用可伸缩大小的滑动窗口提高算法的查全率，减少不必要的比较次数

（3）通过标记含有不完整排序键值的记录，提高了算法的查全率。

IMPN算法的流程如下图3.4所示：



图3.4 IMPN算法流程图

流程图展示的IMPN算法只进行了两趟独立的SNM过程并在最后合并了重复检测结果。设有N趟SNM过程的IMPN算法，则它的步骤如表3.5所示：

表3.5 含有N趟SNM过程的IMPN算法

|  |
| --- |
| **IMPN算法** |
| 1.读取数据集，得到待检测的数据；  2.计算数据集字段的字段区分度并排序；  3.优先选取区分度较大的字段去生成组排序键值；  4.独立地执行步骤5~8次；  5.按照排序键值的产生方式对每条记录提取其排序键值；  6.对数据集按照键值排序，如果某条记录的不完整或者为空则将该记录的ID加入到缺失键值记录集合中，完整则正常排序；  7.进行可伸缩大小的滑动窗口重复检测得到重复集合；  8.将与进行重复归并，然后计算此集合的传递闭包；  9.将N次SNM重复检测得到的集合进行归并，然后计算传递闭包得到最终的重复记录集合。 |

IMPN算法的伪代码如下：

### 3.4.2 算法复杂度分析

时间，空间

## 3.5 实验设计与结果分析

### 3.5.1 实验数据介绍

本章主要是

### 3.5.2 SNM、MPN、IMPN综合对比实验

算法主要以

### 3.5.3 IMPN中对空键值处理方式的验证实验

Logistic

## 3.6 本章小结

# 第四章 改进的遗传神经网络算法

IMPN算法虽然在一定程度上改进了MPN算法，但是它并没有改变本身是基于“排序/归并”去重思想的事实。这种做法一个非常明显的问题就在于，随着数据量的急剧增大，不仅为每条记录生成排序键值以及算法中产生的一些临时重复记录集合会浪费大量空间资源，而且排序过程、滑动窗口归并过程的时间消耗问题也相当严峻。尤其是当数据量过大而不能一次性加载到内存时，需要采用合适的方法将数据集分成簇[5],算法的处理过程变得更加复杂。

另外，传统的“排序、归并”方式在排序完成后需要对滑动窗口内的记录进行相似性检测。常用的方法有字符串的编辑距离法、S-W算法、N-Gram算法等，利用这些算法计算得到两条记录的相似度值，如果大于相似度阈值的话，则断定两条记录相似。这种方式的缺点主要有两个：首先是这些字段的相似度算法往往只对某种特定情况的字段特别有效[52],普适性较差，其次是整条记录的相似度和某字段的相似度并不是简单的线性关系，所以即便是对不同的字段赋予不同的权重，也可能会出现判断失误的情况，况且如何合理地对字段分配权重也是一个复杂的问题，还会导致时间消耗增加。所以理论上最好的方法是通过学习给定的数据集合的特征，利用数据集记录和字段的内在关系来判断记录之间是否重复。ANN（Artificial Neural Network， 人工神经网络）是一种由大量处理单元组成的自适应系统，它的数学计算模型模仿生物神经网络的结构和功能，可以实现对函数的估计或近似。人工神经网络在相似重复记录检测领域也有许多的应用，本文提出了一种改进的遗传神经网络算法，采用BP（Back Propagation， 反向传播）神经网络，通过基于监督的学习实现对训练数据集的相似性关系的非线性模拟，训练完成后即可对测试数据集的相似性进行预测，并通过引入遗传算法克服BP神经网络易陷入局部最优值的缺点。

## 4.1 遗传神经网络用于相似重复记录检测

### 4.1.1 BP神经网络的设计

误差反向传播算法是一种前馈神经网络，它具有学习方式较为简单的优点，应用到结构简单层数少的网络模型中时收敛较快等优点，是一种目前应用最为广泛的有监督学习算法。接下来分析介绍BP神经网络结构的设计思路。

一般情况下的BP神经网络层数多采用3层或者以上，包括输入层、输出层、隐含层，定理[53]指出：三层的反向传播神经网络能够以任意的精度逼近非线性函数。所以设计BP神经网络应当优先考虑增加隐含层节点的个数而不是增加网络层数，网络层数的增加在降低误差的同时也带来了网络复杂度的增加，这会使得训练过程花费的时间大大增加，所以这里的设计采用三层的BP神经网络对相似重复记录进行处理。

输入层的节点数由数据集中记录的字段数目确定，举例来说，如表4.1所示：

表 4.1 数据集部分记录展示

|  |  |  |  |
| --- | --- | --- | --- |
| ID | date\_of\_birth | given\_name | surname |
| rec-409-org | 19881117 | olivia | abat |
| rec-447-org | 19930723 | olivia | quigg |
| … | … | … | … |
| rec-454-dup-0 | 19870221 | sophie | bozdaa |

每条记录的字段数目为4，所以输入层节点数为5（加上偏置）；输出层节点为1，输出的结果介于0到1之间，表示输入向量对应的两条相似重复记录的相似程度；隐含层节点数目的计算由公式4-1[54][55]计算得到。

 公式 4-1

其中代表隐含层节点的个数，代表输入层节点的个数，代表输出层节点的个数，计算过程的结果需要四舍五入进行修整。在本例中，=1，=4，=4，=6，所以计算得到隐含层节点数=6。

激活函数的选择有较多种这里采用常用的Sigmoid函数，即。网络的初始权值的选择较为关键，它影响着网络的学习是否达到局部最小、最终是否能够收敛，甚至包括训练时间的长短。一般最理想的情况是初始化赋值之后，每个神经元的输出接近于0，这样可以在S型激活函数导数最大的地方进行调节与变化，一般取初始值在(-1, 1)之间的随机数。

学习速率决定了每次训练之后权值更新的幅度，过大的学习速率会导致系统震荡，过小的学习速率又会导致训练学习时间变长，收敛速度也变得很慢。一般倾向于选择较小的学习速率，因为这样能保证系统稳定性而不会导致修改幅度过大，取值范围一般在0.01~1之间。期望误差代表着当训练后的误差结果在可接受范围之内则人为进行收敛，停止训练。

表4.1对应例子的3层神经网络的拓扑结构如图4.1：



图4.1 神经网络的拓扑结构

当网络结构确定以后需要提取训练数据集以及网络理想输出对BP神经网络进行训练，训练的目的是为了找到最优情况下的权值以及阈值，通过理想输出和实际输出的误差，然后沿梯度方向修改更新网络权值。使用BP神经网络进行相似重复记录检测的算法流程图如图4.2所示：



图4.2 BP网络检测相似重复记录流程图

其中，停止训练的条件除了上图中的条件（实际输出和目标输出的偏差值在可接受范围之内）外，还可以是达到了人为设定的迭代次数或者权重的更新低于某个阈值。

### 4.1.2 遗传神经网络与重复记录检测

Jaro+tf-idf

遗传改进以及参数配置

步骤、流程等

## 4.2改进的XXX算法

本论文更倾向使用Flume架构来进行开发。

### 4.2.1 算法改进点XXX

本论文更倾向使用Flume架构来进行开发。

### 4.2.2 XXX算法总体设计

本论文更倾向使用Flume架构来进行开发。

## 4.3 对比实验

本论文更倾向使用Flume架构来进行开发。

## 4.4 本章小结

本章主要是描述推荐系统的需求，包括实时性、满意性、惊喜度、多样性、推荐透明和新颖性。具体的阐述了系统的整体上的需求，对需求的每个部分进行解析。同时突出了系统里面的实时性和冷启动问题，另外两种问题只做讨论。

# 第五章 Fork/Join并行框架在IMPN算法中的应用

本章主要是阐述推荐系统的设计和实现，主要包括系统的整体框架，及在框架的基础上系统模块的的设计和模块和模块之间的衔接，最终通过简单的例子阐述系统采用的基于物品的协同过滤算法流程及原理。

## 5.1 整体框架

图5-1实时推荐系统的总体结构示意图

## 5.2 日志采集模块

在该系统上，用户通过点击和浏览会产生一定的行为。如：浏览商品、点击收藏、购买产品等。这些行为都会产生对应的数据保存起来。

### 5.2.1 日志数据类型

日志数据主要包含这几类的日志内容：用户浏览的信息记录，用户检索信息记录、用户收藏信息记录，用户购买信息记录。下述列出应用后台的表设计：

|  |
| --- |
| 12 2:2.5 |

# 第六章 总结与展望

随着“大数据”时代的来临，互联网的信息资源越来越丰富，从而造成信息的过载，导致人们在网上很难容易得到需要的信息。于是，推荐系统的发展对于互联网越发的重要。

本文主要是实现关于电商的实时推荐系统，利用目前比较流行的实时计算框架Storm来实现。前期也尝试过通过流式计算直接实现推荐算法，但是在数据量比较大的情况下，实现的效果并不太理想，所以在后面加入离线平台，把复杂的算法计算放在线下处理，来解决由于计算过于复杂无法及时得到推荐列表的问题，这算是该系统的一个亮点。

该系统包括了系统采集模块、系统接入模块、系统的离线计算、系统流计算模块、系统存储和输出模块。其中，也用简单例子说明算法的计算的过程。由于为了体现实时性，所以该系统上使用比较基础的基于物品的协同过滤算法，目的是为了让线上更新推荐计算不会过于复杂，导致推荐结果输出太慢而影响用户的体验。在文中已经分析了如何通过增量算法更新物品相似度，并利用时间窗口来获取近期的用户行为来进行推荐结果的计算。当然本人也还在尝试利用复合的算法来实现的推荐系统，但是目前效果还是不理想。

由于各种因素的存在，包括时间、环境和本身的能力。本系统还有很大的改进空间。本系统还可以向这几个方向进行探讨：

（1）关于算法方面，可以研究和探讨更适合的算法，不管是复合算法还是新型的算法；本系统只是采用基于物品的协同过滤的算法来实现，结合了在线的更新算法满足推荐效果。但是不同的推荐算法有不同的优势，所以还需要继续试验和探讨。

（2）需要更多的真实商业环境下进行测试工作，目前，这边的数据集，只是从

网上获取的数据用来测试，还是具有片面性。

（3）在离线方面可以利用深度学习，来获取更多有用的信息，以便于在线的推荐。

参考文献

1. 蔡自兴. 人工智能及其应用:研究生用书[M]. 清华大学出版社, 2004.
2. 李勇, 徐振宁, 张维明. Internet个性化信息服务研究综述[J]. 计算机工程与应用, 2002, 38(19):183-188.
3. 项亮. 推荐系统实践[M]. 人民邮电出版社, 2012.
4. Liu F, Tang B, Yuan X, et al. Recommender System in E-commerce[C]// International Conference on E-Business and E-Government. IEEE Computer Society, 2012:700-703.
5. D’Souza S. Apache flume[J]. 2003.
6. Garg N. Apache kafka[M]. Packt Publishing, 2013.
7. Components S, Implementation S. Easy, Real-Time Big Data Analysis Using Storm[J]. Dr Dobbs Journal.
8. Somasundaram N. "Apache Samza - A Stream Processing Framework"[J]. 2014.
9. Akidau T, Balikov A, Bekiro, et al. MillWheel: fault-tolerant stream processing at internet scale[J]. Proceedings of the Vldb Endowment, 2013, 6(11):1033-1044.
10. Neumeyer L, Robbins B, Nair A, et al. S4: Distributed Stream Computing Platform[C]// IEEE International Conference on Data Mining Workshops. IEEE, 2010:170-177.
11. Vavilapalli V K, Murthy A C, Douglas C, et al. Apache Hadoop YARN: yet another resource negotiator[C]// Symposium on Cloud Computing. 2013:5.
12. Shvachko K V. Apache hadoop[J]. Usenix Org.
13. Ghemawat S, Gobioff H, Leung S T. The Google file system[C]// Nineteenth ACM Symposium on Operating Systems Principles. ACM, 2003:29-43.
14. Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters[C]// Conference on Symposium on Opearting Systems Design & Implementation. DBLP, 2004:137-150.
15. Miner D, Shook A. MapReduce Design Patterns: Building Effective Algorithms and Analytics for Hadoop and Other Systems[M]. O'Reilly Media, Inc. 2012.
16. Su X, Khoshgoftaar T M. A survey of collaborative filtering techniques[J]. Advances in Artificial Intelligence, 2009, 2009(12):4.
17. Schafer J B, Konstan J A, Riedl J. E-Commerce Recommendation Applications[J]. Data Mining and Knowledge Discovery, 2001, 5(1):115-153.
18. 张振亚, 王进, 程红梅,等. 基于余弦相似度的文本空间索引方法研究[J]. 计算机科学, 2005, 32(9):160-163.
19. Adler J, Parmryd I. Quantifying colocalization by correlation: the Pearson correlation coefficient is superior to the Mander's overlap coefficient.[J]. Cytometry Part A, 2010, 77(8):733.
20. 孟海东, 张玉英, 宋飞燕. 一种基于加权欧氏距离聚类方法的研究[J]. 计算机应用, 2006, 26(s2):179-180.
21. Fligner M A, Verducci J S, Blower P E. A Modification of the Jaccard–Tanimoto Similarity Index for Diverse Selection of Chemical Compounds Using Binary Strings[J]. Technometrics, 2002, 44(2):110-119.
22. 刘庆鹏, 陈明锐. 优化稀疏数据集提高协同过滤推荐系统质量的方法[J]. 计算机应用, 2012, 32(4):1082-1085.
23. Canny J. Collaborative filtering with privacy via factor analysis[C]// International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM, 2002:238-245.
24. O'Donovan J, Smyth B. Trust in recommender systems[C]// International Conference on Intelligent User Interfaces, January 10-13, 2005, San Diego, California, Usa. DBLP, 2005:167-174.
25. Breese J S, Heckerman D, Kadie C. Empirical analysis of predictive algorithms for collaborative filtering[J]. New Page, 2013, 7(7):43--52.
26. Cannon R K. Scribe: US, US3100346[P]. 1963.
27. 张川, 邓珍荣, 邓星,等. 基于Chukwa的大规模日志智能监测收集方法[J]. 计算机工程与设计, 2014, 35(9):3263-3269.
28. Pazzani M J, Billsus D. Content-based recommendation systems[M]// The adaptive web. Springer-Verlag, 2007:325-341.
29. Cantador I, Bellog, Alejandro N, et al. Content-based recommendation in social tagging systems[C]// ACM Conference on Recommender Systems. ACM, 2010:237-240.
30. Resnick P, Varian H R. Recommender systems. Commun ACM[J]. 1997, 40(3):56–58.
31. 李忠俊, 周启海, 帅青红. 一种基于内容和协同过滤同构化整合的推荐系统模型[J]. 计算机科学, 2009, 36(12):142-145.

致谢

时光荏苒，我的硕士生涯已接进尾声。这几年的时光既漫长又短暂，其中充满了酸甜苦辣，更有收获和成长。几年来，感谢陪我一起度过美好时光的每位尊敬的老师和亲爱的同学，正是你们的帮助，我才能克服困难，正是你们的指导，我才能解决疑惑，直到学业的顺利完成。

本人的学位论文是在我的恩师刘志镜教授和姚勇副教授的殷切关怀和耐心指导下进行并完成的，衷心感谢我的恩师对我的淳淳教诲和悉心关怀。从课题的选择、项目的实施，直至论文的最终完成，他们都始终给予我耐心的指导和支持，我取得的每一点成绩都凝聚着恩师的汗水和心血。他们开阔的视野、严谨的治学态度、精益求精的工作作风，深深地感染和激励着我，在此谨向他们致以衷心的感谢和崇高的敬意。

感谢西安电子科技大学大学 14 级硕士班的全体同学陪我一起走过这段人生难忘的历程!

作者简介

##### 基本情况

朱群，男，福建人，1991年11月出生，西安电子科技大学计算机学院计算机技术专业2014级硕士研究生。

##### 教育背景

2010.09～2014.07 福建师范大学，本科，专业：软件工程

2014.09～2017.07 西安电子科技大学，硕士研究生，专业：计算机技术

##### 攻读硕士学位期间的研究成果

###### 3.1发表学术论文

1. Hong ZHOU, Zhi-jing LIU, Qun Zhu, Bing-hua WANG, Jing-jie LI. The Problem-Solving Agricultural Knowledge Service System for Farm Households, ICAMM, Bangkok Thailand, Octobe 2016, 2016 International Conference on Applied Mathematics and Mechanics, DEStech Publications,Inc. 2016, 472-478.

###### 3.2参与的科研项目

1. “陕勤网”项目，2015-5~2016-07，负责该项目手机Web的开发和维护。
2. 陕西省慧农科技“农掌门”项目，2015-10~2016-04，负责该项目Android APP农民端和专家端的开发和维护。