摘要

摘要

随着信息技术的发展与信息化建设的深入研宄，激烈的市场竞争对于企业信息化 程度的要求也越来越高，在信息发展过程中，数据不断累积，但是原始数据往往是脏 数据，例如错误的、相似重复的和缺失的数据等，其中多源数据归并造成的数据重复 是该领域的热点问题。

本文主要对相似重复记录检测的相关算法进行了研宄与创新。相似重复记录检测 是指准确检测出源数据集中的重复数据，以达到清洗数据的目的。真实情景中，数据 规模庞大，数据来源多样，这都增加了重复数据检测的难度。虽然存在解决这类问题 的优秀算法，例如近邻排序算法和多趟近邻排序算法等，但是已有的算法在解决实际 应用中的重复记录检测问题时，仍存在不足之处。

本文首先研宄了传统的多趟近邻排序算法，并对该算法的缺点进行改进，提出了 改进的多趟近邻排序算法（OMPN)，以适用于实际问题；然后，通过研宄基于遗传 神经网络求解重复检测问题的算法，将OMPN算法与神经网络相结合，得到准确度 更高的A-OMPN算法和A-OMPN算法；最后，将本文提出的OMPN算法应用于“航 天情报信息管理系统”的数据清洗模块，该算法在实际应用中得到了较好的效果。本 文的主要内容如下：

1. 改进的多趟近邻排序算法。多趟近邻排序算法首先对数据集中的数据记录依 据预先选取的排序关键字进行排序，使得相似重复记录排序后位置相近，然后使用滑 动窗口对排序后的数据进行判等。但是，该过程需要依赖专家经验知识进行关键字的 选取和判等字段的选取，也没有考虑真实数据可能存在数据缺失的问题，同时，固定 大小的滑动窗口不仅会导致对重复数据的检测不全面，而且会导致对非重复数据的冗 余检测。本文在多趟近邻排序算法的基础上，提出基于字段区分度的关键字选取方法， 根据数据特点进行关键字的选取，同时，在判等过程中，同样根据字段区分度为字段 赋予不同权值，避免人为因素；然后，采用自适应大小的滑动窗口对排序后的记录进 行检测，减少了漏检记录数量和冗余操作；最后，对源数据中存在缺失值的记录进行 标记和单独检测。通过实验验证，本文所提出的改进的多趟近邻排序算法具有较高的 查全率，且更适用于真实问题场景。
2. 基于神经网络的多趟近邻排序算法。基于遗传神经网络进行相似重复记录检 测的算法效果较好，但是该算法时间复杂度较大，耗时严重。本文将多趟近邻排序算 法与遗传神经网络相结合，提出了基于遗传神经网络的增强的MPN算法，记作 A-OMPN，使得神经网络可以仅对同一个滑动窗口内的记录进行判等，避免了传统的

遗传神经网络对数据全集上的任意两个不同记录进行判等，极大地提高了算法运行效 率。同时，考虑到遗传神经网络训练速度慢的缺点，本文尝试使用单一的神经网络执 行判等操作，得到了基于单一神经网络的多趟近邻排序算法（A-OMPN)。实验结果 表明，本文所提出的这两种算法准确度和运行效率较高。

1. 本文所提出的算法在“航天情报信息管理系统”中的应用。本文主要完成了 该系统的数据清洗模块和APP模块的开发。在真实业务场景中，航天情报管理系统 的数据清洗模块需要实现对源数据的去重和清洗，该系统所使用的数据是真实的不带 标签的数据，且数据规模相对较小，综合分析OMPN算法、A-OMPN算法与A-OMPN 算法的优势与适用场景，最终采用OMPN算法实现系统的数据清洗模块。

关键词：相似重复检测，数据清洗，多趟近邻排序，神经网络，遗传算法

ABSTRACT

ABSTRACT

With the development of the information technology and the information construction, the size of the data becomes larger and larger. Variety of dirty data are inevitable, such as wrong data, reduplicative data and half-baked data and so on. As a result, effective algorithms are necessary for data cleaning. The duplicate records detection problem is one of the most important problem in data cleaning.

In this paper, we have researched and improved the algorithms for the duplicate records detection problem. The duplicate records detection problem is to find the reduplicative records for a given dataset. In real world, it5s difficult to design effective algorithms for the problem since the large size and the different sources of the data. Although there are some algorithms for solving this problem, such as the Sorted-Neighborhood Method (SNM) and the Multi-Pass Sorted-Neighborhood Method (MPN), they all have shortcomings when tackle the real-world duplicate records detection problems.

The effectiveness of the SNM and the MPN relies on the expert knowledge of the dataset. So it’s hard to solve dataset with no priori knowledge. With the goal of overcoming the shortcomings of the SNM and the MPN, we proposed the Optimized Multi-Pass Sorted-Neighborhood Method (OMPN). In addition, we make a combination of the OMPN and the genetic-based artificial neural network to solve the problem and propose the Advanced Multi-Pass Sorted-Neighborhood Method (A-OMPN) and the BP network based Multi-Pass Sorted-Neighborhood Method (A-OMPN). The A-OMPN and the A-OMPN are superior to the other algorithms. Finally, we apply the proposed algorithm to the spaceflight information management system to accomplish the data-cleaning in the real-world problem. The main contributions of this paper are as follows:

1. The Optimized Multi-Pass Sorted-Neighborhood Method (OMPN) is proposed. The MPN first sort all the records and then use a scale-fixed sliding window to check the duplicate records. However, it need the expert knowledge to select the key and to check the duplicate records in a sliding windows. In the OMPN, the field distinction degree-based method is proposed to select the key without the expert knowledge. In the meantime, the OMPN uses the scalable sliding window to make the checking process more precise. The

OMPN also take the half-baked data into account by pre-label scheme. Compared with other algorithms，the OMPN performs well and it’s suitable for solving the real-world duplicate records detection problem.

1. The Advanced Multi-Pass Sorted-Neighborhood Method(A-OMPN) is proposed. The genetic-based artificial neural network that used to solve the problem should select two different records in the whole dataset to check whether they are duplicate or not. It’s very time-consuming and the check stage can be simplified. The A-OMPN makes a combination of the genetic-based artificial neural network and the OMPN to select records only in a sliding windows. It can not only improve the precision ratio and the recall ratio but also reduce the runtime compared with the genetic-based artificial neural network. However， to train an appropriate genetic-based artificial neural network is still time-consuming. We also do experiments with the signal BP network and then generate the BP network based Multi-Pass Sorted-Neighborhood Method(A-OMPN). Experimental results show that the A-OMPN and the A-OMPN all perform well.
2. We apply the proposed algorithm to the spaceflight information management system. The data cleaning module is one of the most important modules in this system. We do experiments by the OMPN，the A-OMPN and the A-OMPN with the given aerospace craft dataset. Finally，we choose the OMPN to accomplish this module.

Keywords: Duplicate Record Detection，Data Cleaning，Sorted-Neighborhood，Neural Network，Genetic Algorithm

第一章绪论

1.1研宄的背景和意义

在现今的信息时代，为了在激烈的市场竞争中占据先机，保险、金融等各种行业 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，数据库的信息量也与日逐增[1]。

从规模庞大的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宄重点。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，从而产生各种类型的“脏 数据”，例如不可避免的数据重复、缺失、错误等如果这些数据不能被正确清洗， 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 借助计算机技术，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，所以研宄的一个重要方向就是尽可能减少人 为的参与和影响[4]。

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宄范畴的一个热点问题[5][6]。

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[5]:

1. 资源浪费：重复记录会带来对存储空间的浪费。
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。它们共同对应的现实中的实体发生变化会导致这些记录 中只有某个或者某些记录发生改变，而其余无法同步更新。

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[6]。

1.2国内外研宄现状

早在上个世纪50年代，数据清洗已经开始了相关研宄。将出自不同数据源的数 据集进行整合被认为是一个困难而且极为重要的问题，最早的研宄主要是从数据连接 [7]、数据实体识别[8]、对象识别等问题来展开，是商业保险、医疗、等领域中的研宄 重心之一。美国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宄的 开端[9]。

数据清洗的研宄重点包括：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[9]。因而相似重复记录的检测与清除成了数据清洗领域的研宄 重点。

在重复记录清洗方面，国外展开了大量的研宄，主要的工作有两个方面一一属性 匹配[10]和重复检测[11]。

相似重复记录检测领域一种主流的算法是“排序/归并”法，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。最基本的算法是“排序/合并”算法[11]。这 种方法有很大的不足，许多研宄人员在此基础上提出了各种各样的改进思路和算法实 现，主要的改进方向包括对字段相似度匹配算法的改进和对相似记录判断方法的改进。

用于属性匹配问题的方法主要有编辑距离算法[12]和递归属性匹配算法[10]等。

Monge等人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[12]。Hernandez等 提出了一种有效地多趟近邻排序算法（Multi-Pass Sorted Neighborhood，记为MPN) [13][14]，该算法的操作过程是对SNM算法（Sorted Neighborhood Method，记为SNM) 独立执行多次，每次采取不同的排序关键字段以及较小的滑动窗口，最后使用C语 言重写的OPS5[15]规则编程判定记录是否相似。SC Hong等提出了基于优先级队列的 方式进行相似重复记录检测[16]。Gianni Costa等人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，解决了大文本库中的相似检测问题[17]。Alfredo Ferro使用了基于q-grams的相似度衡量函数[18]，可以避免许多不必要的比较和判断， 提高了时间效率。

国内的相关研宄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 大学周傲英等比较早开始数据清理的研宄工作[19]。邱越峰等提出了基于N-Gram的相 似记录检测算法[20]。算法以一条数据的N-Gram值作为排序键，该算法在对因为拼写 错误而造成的重复记录进行检测时表现良好。另外，还有基于权重进行相似重复记录 检测的方法[21]，具体实现过程是，首先按照字段的等级划分进行权重赋值，然后结合 长度过滤的思路减少冗余的字段相似度计算。

在数据清洗市场化领域，国内外涌现了一批优秀的数据清洗软件以及框架，包括 商业上和各大学以及研宄机构开发的数据清洗软件[22][23]。

在相似重复记录检测领域，国内外研宄人员已取得了诸多进展，但仍旧或多或少 存在适用局限性或者检测效率和精度不足等问题，所以仍然有研宄的价值和改进的空 间。

1.3论文研宄的主要内容

相似重复记录检测领域的发展虽然已经取得诸多有效成果，但仍旧存在一定问题， 主要体现在[4]:

1. 检测效率与查全率存在提升空间，尤其是较大数据量的相似重复检测问题。
2. 大多数数据清理只针对特定领域以及业务场景，各行业需要更加通用的相 似记录检测方案。
3. 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 研宄的重点问题。

本文在分析常用相似重复记录检测算法的基础上，首先，针对传统的多趟近邻排 序算法MPN在时间消耗和检测精度的不足，提出了改进的OMPN算法。OMPN算 法有三个改进点：（1)通过统计字段区分度改善了传统的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减少被 遗漏的重复记录；（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。其次，在OMPN算法的基础上，本文使用反向传播神经网络对OMPN 算法中的的滑动窗口内的记录进行判等操作。传统的基于遗传神经网络进行重复记录 检测的算法时间复杂度高，需要对所有待处理的数据进行判等，而OMPN算法将数 据全集缩小至可伸缩的滑动窗口内进行判断。在本文中，将两种算法的优势相结合， 提出基遗传神经网络判等的A-OMPN算法和基于BP神经网络判等的A-OMPN算法， 与OMPN算法相比，这两种算法在查准率性能方面得到很大提升。

1.4论文结构

论文一共分为六章，每一章的主要内容如下：

第一章是绪论。主要介绍了的数据清洗研宄的背景和意义，相似重复记录的国内 外的研宄与发展现状，简单描述了论文的主要研宄目的以及研宄内容，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。首先简单介绍了衡量字段相

似度的相似度检测有关算法，分析了它们各自的优缺点以及适用条件。然后介绍了最 基本的近邻排序算法和多趟近邻排序算法，对算法原理、设计以及执行流程和算法的 优缺点都进行了介绍。除此之外还介绍了其它常用算法包括优先级队列算法、N-Gram 算法等。然后对本文用到的BP神经网络理论基础进行说明。最后介绍了相似重复记 录检测领域衡量算法的几个常用标准及其计算方法。

第三章，首先阐述OMPN算法的创新灵感来源，然后详细介绍了 OMPN的操作 思路，并采用SNM算法和MPN算法作为对比，通过实验结果，证明了 OMPN算法 查全率较高的优势，并对算法的优缺点进行了详细分析。

第四章首先介绍了基于遗传神经网络进行相似重度检录检测的算法思路，通过分 析该算法的优缺点，在结合OMPN算法的基础上，提出A-OMPN算法和A-OMPN 算法这两种针对OMPN算法的改进算法，详细介绍了算法思路，并通过实验验证了 算法的较优的性能。

第五章，主要介绍了航天情报信息管理系统中的数据清理模块，该模块是OMPN 算法在该系统中的应用，主要内容包括数据清理模块的设计、重复记录产生的原因、 OMPN算法在系统中的应用方式以及该算法对数据质量的提高等。

第六章，总结了本文的内容以及相似重复记录检测算法研宄过程中遇到的问题， 并对未来的研宄方向进行了展望。

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

相似重复记录是指，数据库中存在这样的两条记录6、尽，它们的内容相同或 者相似，且都对应着同一个现实实体五，则记录对 <尺，牟 > 互为相似重复记录[24]。 实际数据库中可能存在多对互为相似重复的记录，它们的存在降低了数据的质量，可 能会妨碍系统的正常运行，甚至会影响企业信息管理系统的决策正确性。

表2.1给出了学生信息表中的相似重复记录示例：

表2.1学生信息表中的重复记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Stu\_ID | Name | Gender | Brithday Date | School |
| 1801001 | Sam Water | M | 1993/01/02 | School of Computer Science, Xi’an  University of Electronic Science and  Technology |
| 1802002 | Jack Panda | Female | 1990/07/20 | School of Artificial Intelligence,  Xi’an University of Electronic  Science and Technology |
| 1801003 | S. Water | Male | 1993/1/2 | Schol of Computer Science, Xi’an  University of Electronic Science and  Technology |
| 1802004 | Jack Panda | Female | 1990/07/20 | School of Artificial Intelligence,  Xi’an University of Electronic  Science and Technology |
| 1801005 | Mr.Sam W | Male | 1993-01-02 | College of Computer Science, Xi’an  University of Electronic Science and  Technology |

表2.1展示了 5条学生记录，其中Stu\_ID为1802002和1802004的两条记录的 所有字段内容完全一致，说明这两条记录对应现实世界中的同一个学生的信息，所以 它们互为相似重复记录。表中Stu\_ID为1802001、1802003、1802005的三条记录表 面上内容是不一样的，它们的区别在于:Name字段值分别为“Sam Water”、“S. Water”、 “Mr.SamW”，是由“Sam Water”采用了不同的书写方式而产生的；Gender字段值

“Male”、“M”则是全称和缩写的区别，均代指男性；Brithday Date字段则是使用的 不同的时间格式，但它们都是代表相同的一天“1993/01/02”；所属学院字段中，出 现了 “Schol”这样的拼写错误；经过以上观察分析可以发现，这三条内容相似的记 录同样对应同一个学生。

相似重复记录产生的原因多种多样，包括人工操作过程中的录入错误或者管理错 误造成的重复、不同来源的数据集进行合并时产生的重复、信息系统重构时新旧版本 的数据库合并造成的重复等[4][19]。

相似重复记录检测目前应用最广泛的手段是基于“排序/合并”的方法[11][13]：首 先对包含重复记录的数据集进行排序，排序使用的关键字按照某种固定的方式（如某 字段的前三个辅音字母等）从记录的相应字段中提取，排序之后相似重复的记录汇聚 在相邻的位置，然后通过对相邻位置的记录进行对比判等，可以检测出相似重复记录。

2.2相似度匹配算法

相似重复记录检测过程中需要对不同的记录对进行整体相似性判断，这就需要用 到字段相似度匹配算法。数据库中的每条记录均由不同的字段组成，相似重复记录各 字段内容也相似重复，通过计算不同记录字段之间的相似度可以判定记录是否互为重 复。目前该领域的算法主要有两大类：基于单字段的匹配算法和基于多字段的匹配算 法。

2.2.1基于单字段的相似度匹配

基于单字段的相似度匹配算法在相似重复记录检测中的应用过程是，通过计算两 条记录相同字段对应内容的相似度来衡量记录整体的相似程度，这是一个从部分到整 体的过程。编辑距离算法[25]、Smith-Waterman算法[26]、Jaro算法[27]等都是用于这类 问题的有效算法。

编辑距离算法是Levenshtein于1965年提出的一种基于字符的相似度匹配算法， 又名L-距离算法[25]。两个字符串&和&的编辑距离是指：&变成&需要对其单个字 符进行插入、替换、删除操作的次数。编辑距离越小代表X和&越相似[25]。

如图2.1所示，字符串“change”经过3次插入操作和一次删除操作可以变成字 符串“challenge”，所以这两个字段的编辑距离为4。计算两个字符串间的L距离的 经典解法是使用动态规划方法。L-距离算法在应对字母书写错误、缩写等场景下效果 较好。

Smith-Waterman算法（简记S-W算法）[26]最早是在生物学序列比对领域被提出 的，是一种用于匹配遗传序列的动态规划算法。它的主要思路是通过罚分和空位计算 不同字段内容的相似度。S-W算法可以有效应对包含不正确值的相似重复记录，但处 理字符串缩写、字母颠倒情况的能力较差。

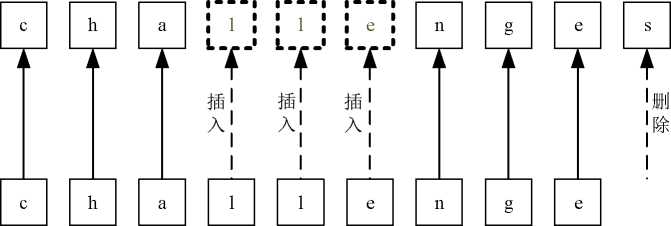


图2.1编辑距离示意图

Jaro算法[27]由Jaro在1976年提出的基于字符串公共子集的相似度匹配算法。Jaro 距离用来衡量两个字符串的相似度，对于给定的字符串乂和&，两者的Jaro距离如 下公式(2\_1)所示[27]：

(2\_1)

在公式(2\_1)中，m代表匹配的字符个数，t代表换位的数目。Winkler提出的 Jaro-Winkler相似度匹配算法在Jaro算法的基础上，赋予相同的字符串更高的分数， 减小了原算法对于字符距离限制的影响，提高了算法在面对较分散的长字符串时的检

测准度。Jaro-Winkler距离的计算如公式(2-2)所示:

*D*J—W = *D*jaro + *L\* P(1*- *D*jaro) (2-2)

在公式(2-2)中，是Jaro距离，L是前缀的匹配长度，P是一个常数，作用 是可以调整前缀匹配的权值，P <0.25。

2.2.2基于多字段的相似度匹配

基于多字段的相似度匹配算法的思想是将一条记录视为一个整体，通过计算两条 记录整体上的相似度判断是否互为相似重复记录。常用的算法包括余弦相似度匹配算 法、基于监督训练的机器学习方法等[28]。

余弦相似度[28]是一种基于词频-逆文档（Term Frequency-Inverse Document Frequency，记为TF-IDF)加权算法的多字段相似度匹配方法。算法的步骤如算法2.1 所示：

算法2.1:余弦相似度匹配算法

1. 将需要匹配的字段内容进行分词，得到互相独立的单词w,〇+ = 0，1，\_\_.«);
2. 对每个单词\¥;分配权重，胃其中单词出现的次数（词 频）用^"Wi表示，表示记录总数除以包含以;的记录个数（逆文档频率）；
3. 将待匹配的字段转化成向量a和6 ;
4. 计算向量的余弦相似度：cos沒:

EI/a.xb.)

1. cos沒值越接近1，则说明相似度越高，将cos沒与所给阈值相比较，判断记录的相似性。

除此之外，机器学习领域中的分类技术可以用来检测判断重复记录[29]。使用基于 单个字段的相似度匹配算法进行相似重复记录检测时，不同的字段拥有不同的判等权 重，对记录相似与否的影响程度也不同，字段之间相似到记录整体的相似度关系是非 线性的，这是一种适合使用基于训练样本的有监督学习的场景。因此，存在使用神经 网络进行判定的方法，通过带标签的数据集（可以明确不同记录之间相似与否）对神 经网络进行训练，然后采用训练好的网络对由记录对生成的输入向量进行计算，相当 于将重复记录检测问题转化为二类分类问题，如果网络输出的结果大于给定的判别阈 值，则判定为重复记录，否则不重复[29]。

2.3相似重复记录检测算法

相似重复记录检测领域最直接的方法是对数据集中的数据进行一对一地判等，这 种做法简单，查重效果好，但是时间复杂度为0(#2)，当数据量比较大时效率过低。 “排序/归并”是目前相似重复记录检测算法所采用的主要思想，其主要过程是：首 先选取排序关键字，关键字可以使预先设定的，也可以按照相应算法进行计算;然后， 对数据集按照所选取的关键字进行排序，将相似记录汇聚到邻近位置；最后，使用相 似度匹配算法进行重复检测。常见的基于“排序/归并”思想算法有近邻排序算法 (Sorted-Nighborhood Method，记为 SNM) [11]、多趟近邻排序算法（Multi-Pass Sorted-Neighborhood，记为 MPN) [13]、优先队列算法（Priority Queue Strategy,记为 PQS) [16]、N-Gram 算法[20]等。

2.3.1近邻排序算法

SNM算法的设计思路是：首先，根据数据领域的专家知识经验指定数据集排序 所使用的关键字的生成方式；其次，遍历数据集对每一条记录生成排序关键字并附加 到记录后，设第i条记录R的排序关键字为fey;然后，按照记录的fey.对记录进行 排序，根据相似记录对应的关键字内容也相似的原理，不同的重复记录在排序完成后理论上会处于邻近的位置；最后采用固定大小的滑动窗口方式对数据集进行重复检测。 SNM算法步骤如算法2.2所示：

算法2.2: SNM算法

1. 确认排序关键字的生成方案，滑动窗口大小为w;
2. 对每条记录&生成排序关键字Aey ;
3. 按照々ey对数据集的记录进行排序（只考虑内部排序）;
4. 对同一个滑动窗口内的记录进行判断。

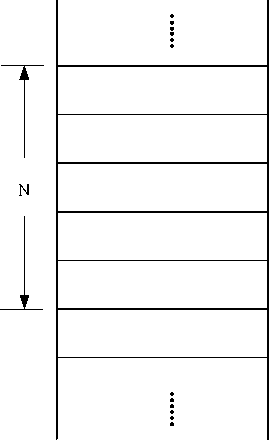
SNM算法滑动窗口的过程如图2.3所示:

图2.2 SNM算法滑动窗口过程示意图

下一个窗口的记录

A N ▼

当前窗口的记录



若数据集大小为#，使用SNM算法生成排序关键字过程的时间复杂度为0(1)， 本文使用快速排序，其时间复杂度为对滑动窗口内记录的判等操作的时 间复杂度为〇(^)，其中w为窗口的固定大小。SNM算法的优点在于使用简单，滑 动窗口判重过程效率较高，运行速度较快。但它也存在比较明显的缺点：

1. 过于依赖生成的排序关键字。选择不当的关键字生成方案可能导致相似重 复记录相距较远，不相似的记录却处于邻近位置，这就导致算法的检测效果大打折扣。
2. 滑动窗口的大小w较难选择。若w太大，虽然检测效果可能提高，但是会 导致算法的运行时间增大；若w太小，很可能导致相似重复记录无法被窗口覆盖到， 导致算法查全率下降。

2.3.2多趟近邻排序算法

MPN算法是在SNM算法的基础上的一种改进算法。该算法的改进点在于：

1. 对数据集互不干扰地执行多趟近邻排序算法，每次采用不同的排序关键字 生成方案，并且滑动窗口的大小相对于传统的SNM算法可以更小。
2. 对执行完多趟SNM算法的结果求传递闭包。

传递闭包的定义是：设A是定义在集合A上的关系，P代表传递性，满足下 列所有条件的关系A称为A的传递闭包：

1. R ^ R ；
2. A满足性质尸；
3. 如果存在集合A上的关系R'，R'满足性质尸并且&'2&，则尺口尺'。

传递闭包的计算多采用Warshall算法[31]，算法2.3给出了 Warshall算法的伪代码:

算法 2.3: Warshall 算法

1. W :=M*r*
2. FOR 1 to n

FOR i := 1 to n FORj := 1 to n

W[i, j] *^* W[i, j] v (W[i,Ik] a W*[*k*,* j])

3.Output W；

MPN算法对多趟检测结果可以计算传递闭包的理论基础是相等的传递性[14]:若 记录R1和R2是相似重复记录，记录R2和尺3是相似重复记录，则R^PR3也互为相似 重复记录。

MPN算法的步骤如算法2.4所示：

算法2.4: MPN算法

1. 确认w个排序关键字的生成方案，独立重复地执行步骤2〜4w次；
2. 对数据集生成排序关键字；
3. 按照= 1，…，对数据集的记录进行排序（只考虑内部排序）；
4. 确定滑动窗口的大小％〇+ = 1，…，，每次比较时将新进入窗口的记录< 与窗口内剩余 的w -1条记录进行相似性判断得到重复记录集合D# = 1，…，w);
5. 对w个重复记录集合求传递闭包得到最终的重复记录集合。

含有两趟SNM过程的MPN算法的流程图如图2.3所示:

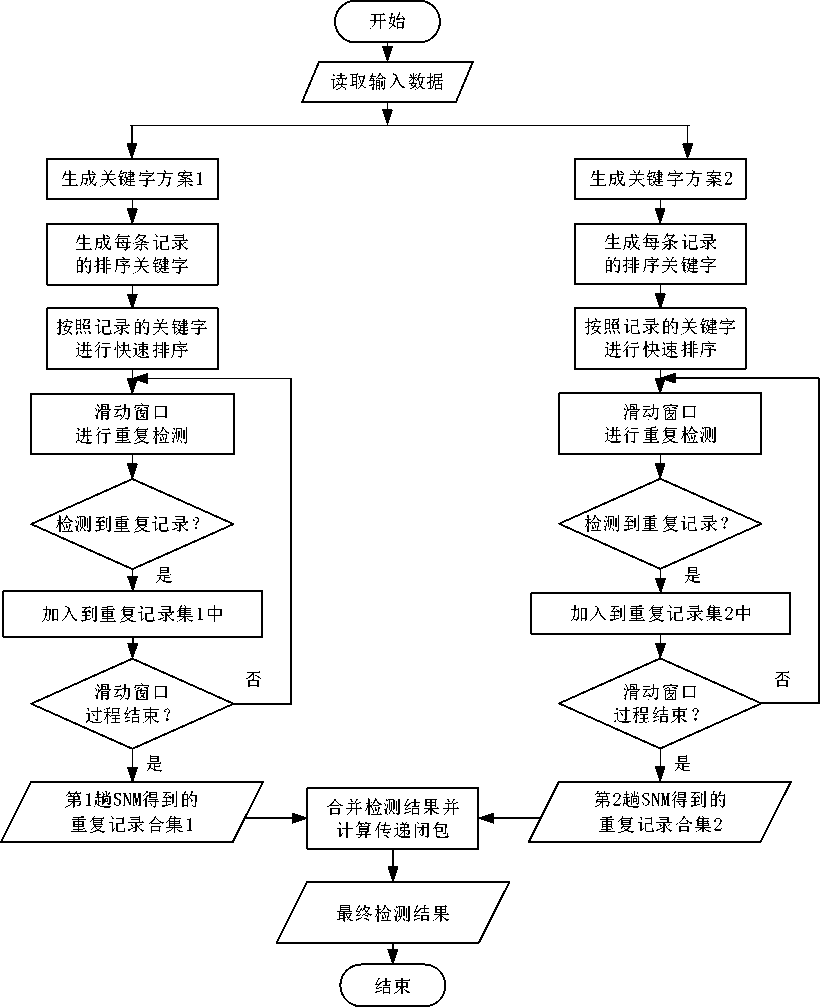


图2.3 MPN算法流程图

正是由于MPN算法引入了传递闭包的计算，使得一些容易被遗漏的重复记录被 检测了出来，提高了算法的查全率，同时每轮SNM过程的滑动窗口也可以变得更小， 缩短了滑动归并的执行时间。但是MPN算法也存在缺点：依旧没有克服SNM算法 对于排序关键字的依赖性，计算传递闭包容易导致算法的误识别率上升。

2.3.3其它算法

N-Gram算法是一种基于聚类思想的算法[32]。N-Gram值由记录中每个单词出现

的概率综合计算得出，相似重复记录的N-Gram值在数值上也相似，算法的优势在于 对常见拼写错误表现较好。韩京宇等人提出了一种基于N-Gram层次空间的聚类算法

DGHS[33]，记录私、尽的N-Gram相似性计算方法如公式(2-3)所示[20]：

*Gq(Ri)^Gq(R)*

(2-3)

G*q*(R*i*)uG*q*(R)

公式(2-3)中,s+mdR)表示记录R、R的N-Gram相似性，G?(R)表示记录R的 所有字段组成的集合。

优先级队列算法进行相似重复记录检测的思想是：用含有不同重复记录簇的优先 级队列来替换传统SNM算法中的滑动窗口，算法在扫描过程中，若队列中不含有当 前记录则赋予其最高的优先级然后加入到队列中，如果含有该记录，则将相应的重复 记录簇的优先级设为最大。采用多趟优先级队列算法进行相似重复记录检测的过程如 图2.4所示[16]：

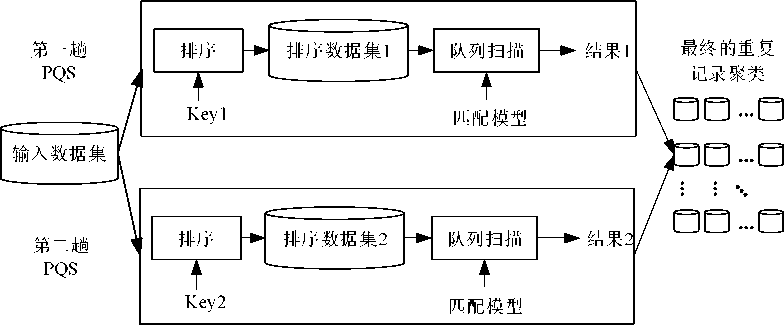


图2.4多趟优先队列扫描算法过程示意图

1. BP神经网络理论基础 2.4.1神经元模型

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[34]，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，从而可以表征真实社会中更加复杂的问题。神经网 络中的一个神经元所起到的作用是接收来自其他神经元的加权输入，然后结合自身的 阈值(偏置)，最后经过非线性函数的处理，得到输出结果[35]。典型的神经元模型如图

2.5 所示[34]:

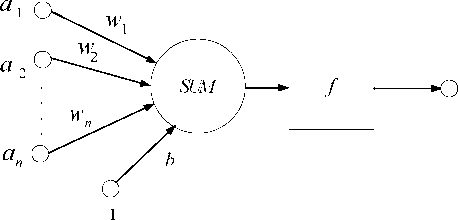


图2.5神经元模型

其中，％是输入向量的第Z个分量，％是第Z个输入分量连接到该神经元的权值，

6表示偏置，是加权求和操作，/表示激活函数。则对于输入向量 /印故=(七化…八）经过此神经元时，经过加权求和以及激活函数得到的输出为：

n

ow炉故=/(^%%+6)。许多的类似于这样的神经元则组成了人工神经网络。

i=1

2.4.2梯度下降法

梯度下降法[36]是一种经典的最优化方法，其主要思想是不断沿着负梯度方向进行 搜索。给定目标函数如公式(2-4)所示，{60，0，...，6^是要学习的参数，< 是第个输

入特征向量的第i个分量，•^ = 1, %表示偏置，共有<i维特征。

d

he( X ) = °0 + ^ixj + °2 X2 +... + edXd = (2-4)

*i*=0

采用均方误差损失函数，如公式(2-5)所示[36]，共有rn个训练样本，y表示第j+个 训练样本的真实类标向量，&(xj)表示第j+个训练样本的预测类标向量，当损失函数 «/(沒)的值最小时，说明所训练出的模型参数最能拟合训练样本，因此求解参数的过 程就是最小化损失函数八内[36]。

1. m 2

J ⑷)=：；艺(尸—W)) (2-5)

1. d=i

首先求出损失函数对参数的导数，如公式(2-6)所示。然后，根据损失函数J⑷)对 参数0的负梯度方向更新参数0 ,如公式(2-6)所示。^是梯度下降法的学习速率，一 般情况下，随着学习次数的增加，参数7逐渐减小，即参数在学习过程中的变化越来 越小。

由公式(2-7)可以看出，对每一个参数6i(/ = 1，2，..，d)，都需要使用全部样本来学

*dJ (&)*

=-±(y*=* -±{y *-h^iyxi*

戶 l j=\

(2-6)

v

8J (&)、

m

0,+ r/T,{yJ ~he(xJ))'x!

(2-7)

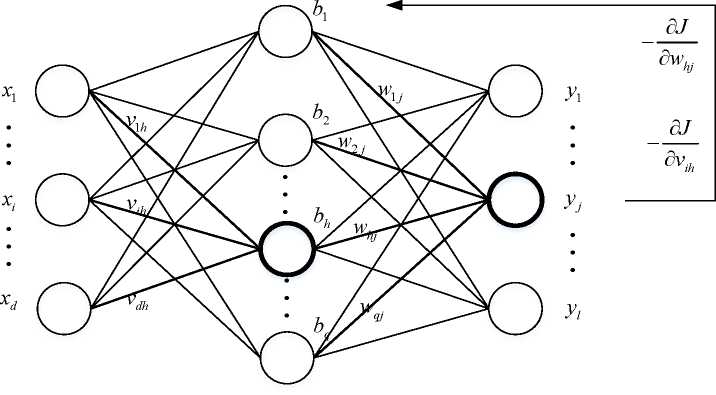


习该参数的变化量，将这种梯度下降法的实现方式称作“批梯度下降法” 37。在实际 操作中，由于样本个数m较大，所以这种参数更新方法会使导致训练过程缓慢，难以 应用于实际问题。

为了克服批梯度下降法的缺点而出现了随机梯度下降法和小批量梯度下降法，这 两种方法使用样本全集中的一个或部分样本来更新参数，这样操作使得每一次并不是 按照严格意义上的最优方向来更新参数，但是从整体来看，依旧是朝着负梯度的方向 更新参数，这两种梯度更新方式使参数学习的速度大大提高，适用于大规模训练样本 的情况。

2.4.3 BP网络前向传播和反向传播

反向传播



ah = z vihxi *b==f* =Z whjbh

i=0 h=0

前向传播

y

图2.6典型三层BP神经网络

一个典型的三层BP网络如图2.6所示，第一层共有d个输入神经元，是输入层； 中间9个隐层神经元构成第二层，第三层包含/个输出神经元，表示输出层[38,39,40,41]。 &表示第i个输入神经元与第h个隐层神经元的连接权值，vM表示第h个隐层神经元

的偏置，表示第A个隐层神经元与第\_/个输出神经元的连接权值，表示第\_/个 输出神经元的偏置，^表示第/个输入神经元的输入特征值，\表示第A个隐层神经 元的输出，I表示第\_/个输出神经元的输出，A表示第A个隐层神经元的输入，&.表 示第\_/个输出神经元的输入。

BP网络的前向传播是指将上一层的输出与层间的对应权值相乘并求和，最终使 用函数/«进行处理，得到下一层对应神经元的输出值，/«代表非线性函数。常用 的非线性函数有sigmoid函数(如公式(2-8)所示)、tant函数(如公式(2-9)所示)、ReLU 函数(如公式2-10所示)等[39’40’41]。

|  |  |  |
| --- | --- | --- |
| f⑷ | \_ 1 | (2-8) |
| =1 + e\_z |
| m: | 1-e-2z | (2-9) |
| =1+e-2z |
| f (z)= | max(0, z) | (2-10) |

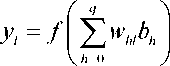
如图2.6所示，前向传播过程如公式(2-11)和公式(2-12)所示，设/(•)使用sigmoid

函数。

bh = f I Z 'vihxi

(2-11)

(2-12)



对于第左个训练样本(4/)二匕=(1:^,4,...,4)，/=(尤\_^,...,乃”，设神经 网络对于该样本的输出为;7)，使用均方误差损失函数，则对样本 (x\/)的损失函数如公式(2-13)所示，样本全集的损失函数如公式(2-14)所示，其中 第一项表示在所有样本上的平均损失，第二项表示12正则项，^表示所有参数组成 的向量，正则项可以避免过拟合。Sigmoid函数对参数求导结果如公式(2-15)所示。 BP网络反向传播的目的是求得使损失函数最小时的参数，使用梯度下降法进行求解。

Jk = 2Z(yl.- ~ yk}) (2-13)

(2-14) (2-15)

根据链式求导法则[36]，损失函数人对参数的导数分别如公式(2-16)和公 式(2-17)所示。

(2-16)

*h*

t (^ - - /(/?；))^ j\*/K) (1 - fM)^ (2-17)

bh(1 - bh) ^ Z (yk} - yk}) yk} (1 - yk})

在样本(分,/)上，参数的更新如公式(2-18)所示。

*whj ^ whj*

*dw*

*hj*

(2-18)

vih ^ vih -n

BP误差反向传播算法的基本过程为：

输入：训练集^〇 = {(4,八)^=1，学习率 输出：连接权值与偏置值确定的多层前馈神经网络.

过程：

产生(0,1)范围内的随机数，初始化所有连接权值和偏置值；

Repeat

F〇r心（及,凡）』：

根据当前参数执行前馈传播，计算当前样本的输出值y1; 根据公式(2.17)进行反向传播，更新当前参数；

End for

Until达到终止条件

2.5本章小结

本章首先介绍了相似重复记录的概念以及产生的原因，然后介绍了常用的基于单 字段和多字段的相似度匹配算法。

然后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、优缺点等对SNM算法、MPN算法等进行了介绍说明。

最后介绍了基于神经网络进行相似重复记录检测的理论基础，包括神经元模型、 梯度下降法、BP网络的传播过程。

第三章改进的OMPN算法

MPN算法在传统的SNM算法的基础上，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 且通过计算传递闭包可以检测到一些人工难以发现的重复记录。

但是MPN算法并没有克服SNM算法在选取排序关键字时过于依赖专家经验的 缺陷，同时，如果某一条记录的排序关键字字段为缺失值时，MPN算法的检测效果 很差。基于MPN算法的不足，本文在MPN算法基础上提出了一种优化的MPN算法 (Optimized Multi-Pass Sorted Neighbothood，OMPN)。OMPN 对 MPN 的改进在于三 个方面，首先，OMPN采用基于统计规律的提取关键字的关键字选取方法；其次， OMPN采用动态可伸缩的自适应滑动窗口；最后，OMPN增加了对排序关键字有缺 失的记录的特殊处理。

3.1基于字段区分度提取关键字的方法

3.1.1传统的提取关键字的方法

基于“排序、合并”思想的重复记录检测算法，都需要依据预先提取排序关键字， 然后依据提取的排序关键字对记录进行排序[42]。在完全理想的状态下，重复记录会具 有相同的关键字，所以，这些重复记录排序后会汇聚到邻近的位置，从而可以缩小检 测范围，在小邻域内进行重复记录的检测。

在相似重复记录检测问题上，排序关键字是指，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[43]。对于传统的SNM算法和MPN算法，关键字的 选取十分关键，数据集中的记录的排序主要依赖于所选取的关键字，只有选择了恰当 的关键字，才能保证排序后的重复记录聚集在相邻的位置，进而保证重复记录能够被 一个滑动窗口所覆盖，从而准确地检测到重复记录。一方面，准确的排序关键字可以 保证较小的滑动窗口就具有较大的检测准确度，另一方面，准确的排序关键字会使得 不同的记录经排序后处于较远的位置，从而可以避免无意义的判等比较，节约检测时 间。综合来看，排序关键字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。

排序关键字的选取方式不是唯一的，传统的SNM算法和MPN算法需要根据专 家经验，人为地为待处理的数据集提取准确的排序关键字，这也是这类算法的一个缺 点。所以，针对特定的数据集，准确地选取合适的排序关键字，才能达到准确检测重 复记录的目的，接下来，通过表3.1所示的内容说明排序关键字的选取对重复记录检

测的影响。

表3.1四条相似重复记录的例子

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Record | First Name | Last Name | Address | ID |
| 尺1 | Jack | Stolfo | 123 First Street | 12345678 |
| 尽 | Jack | Stolfo | 123 First Street | 12345673 |
| 尺3 | Jack | Stolpho | 123 First Street | 12345678 |
|  | Jacon | Stiles | 123 Forest Street | 12345432 |

表3.1列举了四条记录私、2、尺3和凡，它们由四个字段组成，分别是First Name、 Last Name、Address和ID,其中，尺t、八、尺3互为重复记录，&与其他记录不同。 记录八和尺t的First Name、Last Name和Address字段都完全一致，ID字段因人为录 入或印刷错误而不同，但是他们是重复字段。尽和私的Last Name字段不一致，这 也是由于人为错误导致的。而和R的Last Name字段完全不一致，而且两者的 Address字段的取值也相差甚远，它们对应着现实世界的两个不同的实体。相似重复 检测算法就是要从已有的脏数据集中，准确地检测到重复数据，例如，对于表3.1所 示的4条记录，就需要算法准确得到尺=& =馬#&4的结果。

对于表3.1所示的数据，依据传统的关键字选取方法，因为主要依赖专家经验， 容易因人而异，本文列举三种不同的排序关键字选取方式，然后，详细阐述每种选取 方法所得到的排序关键字，并分析不同的排序关键字对检测结果的影响。

方式1、排序关键字的构成包含以下几个部分：

1. LastName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

方式2、排序关键字的构成包含以下几个部分：

1. LastName的前三个辅音字母
2. FirstName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

方式3、排序关键字的构成包含以下几个部分：

( 1) Last Name 的前三个辅音字母

( 2) First Name 的前三个辅音字母

1. Address 的数字部分加字母部分的前三个辅音字母
2. ID 的前三个连续数字

由方式1、方式2和方式3得到的针对表3.1所示数据的排序关键字分别如表3.2- 表3.4所示。

表3.2相似重复记录及其生成的关键字

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 表 3.3 相 | | | 似重复记录及其生成的关键字 | | |
| Record | First Name | Last Name | Address | ID | Key |
| R | Jack | Stolfo | 123 First Street | 12345678 | STLJAC123FRT213 |
| R | Jack | Stolfo | 123 First Street | 12345673 | STLJAC123FRT123 |
| R | Jack | Stolpho | 123 First Street | 12345678 | STLJAC123FRT123 |
| R | Jacon | Stiles | 123 Forest Street | 12345432 | STLJAC123FRT123 |

表3.4相似重复记录及其生成的关键字

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

由表3.2可以看出，记录^和尽对应的关键字相同，均为“STOLFO123FRT123”， 记录R和&对应的关键字不同，经过排序后R和尽会聚集在邻近位置，而R和&则 不会处于邻近的位置。对于记录R和及，它们原本是相同的记录，但是依据这种方 式对它们所选取的排序关键字是不同的，所以将导致无法准确地检测出记录$和及 是一对重复记录。

由表3.3可以看出，按照方式2对4条记录所选取的排序关键字均为 “STLJAC123FRT213”，然而从表3.1和表3.3可以看出：R和尽具有相同排序关键 字是合理的；R和及对应的现实实体是一致的，选择辅音字母可以一定程度上克服 发音上的错误，R和及的排序关键字的相同也是合理的；然而，&和R则是对应着

现实世界的两个不同的实体，由于“Stolfo”和“Stiles”的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”，并且关键字段 中所选取的ID也是相同的，所以导致&和R具有相同的排序关键字，经排序后，& 和将会与A、尽、尺3处于近邻的位置，从而会导致额外的判等计算，甚至可能导致 错误地将K与R、尽、及识别为重复记录，从而影响算法的准确性。

由表3.4可以看出，R、尽、及的排序关键字相同，它们对应的现实实体也是 一致的，&和R则对应不同的排序关键字，这说明所选取的排序关键字可以有效地 将重复数据聚集，将不同数据隔离，因此，这是针对该数据集的理想的排序关键字选 取方式。

通过以上分析可以发现，排序关键字的选取对这类基于“归并/排序”思想的算 法的影响较大。因此需要有一种准确的、不受人为干扰的、基于数据特征的关键字选 取方法。

3.1.2改进的字段区分度方法

衡量排序关键字优劣的一条重要的原则是，不同的记录对应的关键字不同，相同 的记录对应的关键字理应相似或一致[19]。对于真实重复的记录，无论排序关键字以何 种方式选取，理论上会得到相同或近似相同的排序关键字值，所以衡量排序关键字最 关键的标准是它们在区分不同记录时的表现，即对应着现实世界中不同实体的两条记 录在合适的排序关键字方案下生成的排序关键字值应当不一致。所以在选择排序关键 字时，应当选择有足够区分度的字段来提取关键字。为定量地衡量关键字的区分不同 记录的能力，本文提出了“字段区分度”的概念。

字段区分度：字段区分数据库中不同记录的能力，某一字段取不同值的记录个数 越多，该字段的字段区分度越大。

设数据库中一共有#条记录，每条记录都M个字段组成，即 = ...，X}，对于第/个字段灰从，它的区分度计算公式如公

式(3-1)所示：

*diff*field*i=*f (3-1)

其中，xfe/d代表数据集中在fe/d字段一共有xfe/d种取值，即如果将数据集按 照字段fe/d的不同取值进行聚类，一共有xfe/d簇。dffe/d的取值介于0到1之间， dffe/d值越高，说明对应的字段fe/d对于整体数据集的区分能力越大。

通过计算字段区分度，可以有效地衡量不同字段对整体数据集的区分能力，这是

因为，相似重复记录在每个字段的内容理论上是相等的，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的情况在数据集中只占有 较小的比例，因此，无论数据集中的重复数据所占的比例是多少，对数据集具有较大 区分能力的字段的“字段区分度”值也较大。基于字段区分度的排序关键字选取方法 的主要操作过程如算法3.1： 算法3.1：基于字段区分度的排序关键字选取算法

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. 确定排序关键字由w个组成部分，以及每个部分的生成方案= 1，...，w ；
4. 对数据集中的每条记录优先选取区分度较大的字段，按照生成有m个组成部分的排 序关键字{知外，知少2，...知1} ；

本文提出的基于字段区分度的关键字选择方法，依靠已知数据集的统计特性进行 关键字选择，克服了 MPN算法依赖专家经验进行人工选择排序关键字的缺陷，同时， 对于未知类型的数据集，人工选择关键字往往不能准确把握该类数据集的特征，人为 因素对算法效果的干扰较大，而基于字段区分度的方法从统计角度出发，更能挖掘数 据本身的特性和规律，人为因素影响较小。因此，在没有人工参与或者较少人工参与 选取排序关键字的情况下，基于字段区分度选择排序关键字的方法更有利于在排序后 将对应着不同实体的记录区分开，将对应着相同实体的重复记录聚集到相邻位置，从 而提高了算法的普适性。

“排序/归并”类算法在滑动窗口判重过程中，需要对邻近位置的记录进行相似 判断，传统的MPN算法采用的是基于专家经验知识的规则产生式系统（OPS5)，这 种做法依赖人工，效率较低，适用范围较为局限。本文在提出字段区分度的基础上， 进一步提出了基于字段区分度加权的判等方法，在判等过程中，为区分度高的字段分 配更大的权值，字段相似度匹配采用编辑距离算法。对两条记录R、尽（均包含《^^ 个字段）进行基于字段区分度加权判等的具体操作过程为：

1. 对于记录记录R、怂，计算它们之间的在每一个字段下的编辑距离；
2. 按照字段区分度由小到大，对字段进行排序，并对记录的编辑距离根据字 段的相应位置进行调整，得到记录R、怂的调整后的编辑距离向量edY；

(3 )按照公式(3-2)计算赋给第+个字段的权值we+g以；

*weight*

i2

(3-2)

(4)按照公式(3-3)计算记录尺丨、八的相似度幻+rn;

num

sim *= [* (edit x we*+*gh*^* ) (3-3)

i*=*i

1. 将sim和阈值r进行比较，若sim^T，则两条记录为相似重复记录。否则 判定记录不重复。

算法3.2给出了对待判定的记录集进行判等操作的步骤。

算法3.2:基于字段区分度的加权判等算法

1. 设记录集合中不同的两个记录尺i和&，计算记录尺i和&的编辑距离向量edit;
2. 根据字段区分度对edit中的元素进行位置调整；
3. 根据公式(3-2)计算每个字段的权值weight,;
4. 计算尺丨和八的相似度sim ;
5. 将sim与给定阈值进行判断；
6. 对数据集中任意两个不同的记录执行步骤1-5,完成对数据全集的判等操作。

3.2自适应大小的滑动窗口检测方法

MPN算法和SNM算法采用相同的滑动窗口选择方法。算法在操作过程中所采用 的滑动窗口的大小都是一个固定值，记窗口大小为w，本文在此基础上，提出了自适 应大小的滑动窗口方法。

3.2.1传统的滑动窗口检测方法

给定大小固定为w的滑动窗口，首先，将数据集中的所有记录合并成一个线性序 列，然后，将已知滑动窗口从第一条记录开始，每次向下滑动一条记录，每当滑动到 一条新纪录时，移除原来窗口中的第一个记录，始终维持窗口内有w条记录，直到数 据集中的最后一条记录进入窗口内。

基于滑动窗口进行重复记录检测的主要操作过程为：初始时，取得滑动窗口内的 所有w条记录，将这w条记录两两之间进行判断，检测任意两个记录是否是重复记录; 然后，滑动窗口每次滑动所访问到的最新的一条记录分别与之前的w-1条记录进行 比较，判断与这w-1条记录中的任意一条是否相同。

传统的滑动窗口扫描过程如图3.1所示。

图3.1滑动窗口扫描过程

当前窗口的记录

f

下一个窗口的记录

A W T

基于固定大小的滑动窗口检测方法需要提前设定窗口大小，所以人为因素干扰较 大，其次，检测结果对窗口大小的依赖性较大，同时，对于大小为w的滑动窗口，这 种检测过程的时间复杂度为0(^)，所以，如果滑动窗口过大，则会增加算法的运 行时间，如果窗口过小，则会降低算法检测的精度。

图3.2给出了不同窗口大小时的检测结果示意图。在图3.2中，record0\_origin和 record2\_origin 是两条不重复的原始记录，record0\_dup0、record0\_dup1、record0\_dup2 是与record0\_origin重复的原始记录，分别用标号(1)-(5)简单标记这些记录。

图3.2中分别展示了两种大小的滑动窗口，一种窗口大小为2，一种窗口大小为 5。若滑动窗口 w的大小为2,按照传统的滑动窗口归并方法，无法检测到记录(4)与 记录(1)、记录(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，但是检测时间是第一种检测时间的2.5倍。

通过观察可以得到，记录(1)、记录(2)相邻且是重复记录，所以应当采用较大的 滑动窗口，以对尽可能多的记录进行检测，而当窗口移动到记录(3)、（4)、（5)位置的 时候，这些记录互相不重复，较大的滑动窗口只会增加检测时间，所以适合采用较小 的滑动窗口进行检测，既不会降低检测精度，又可以节约时间。通过以上分析可以得 出，SNM和MPN所采用的固定大小的滑动窗口方式仍有较大的优化空间，一个较为 合理的滑动窗口检测方案应当可以根据数据内容动态变化滑动窗口的大小，以减小算 法复杂度。

图3.2不同滑动窗口大小时的归并过程

a

|  |  |
| --- | --- |
| (1) | recordQ origin |
|  | |
| (2) | recordO dupO |
|  | |
| (3) | record1 origin |
|  | |
| (4) | record0 dup2 |
|  | |
| (5) | record2 origin |

w2 = 5



W = 2



3.2.2改进的自适应大小的滑动窗口

为克服传统的固定大小的滑动窗口的不足，本文提出了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，证明当前窗口正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，应当扩大窗口尺寸以使 得窗口包含更多的记录，对更多的记录进行检测；反之，如果滑动窗口内的数据集重 复度比较低，即相似重复数据较少，则说明当前位置附近的数据之间可能互相不重复， 会存在冗余的检测过程，所以如果窗口较大，就会造成这些互不重复的数据需要不断 与新进入窗口的数据进行重复性检测之后才能退出滑动窗口，导致运行时间的增加， 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[44]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，其中W是滑动窗口的大小， #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。

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。也就是说，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。本文提出了动态计算滑动窗口大小的计算公式，如 公式(3-2)所示：

index*-1* ^ \i *-* index*\x*

*=*ind*^*x*-*w*+1*

(3-4)

Wnext = Wmm + (Wmax - Wmm)：

index-1

^ |i - inde:x\

i*=*ind*^*x*-*w*+1*

公式(3-2)中，常数％\_表示滑动窗口的大小可以取的最大值，常数Fmn表示滑动 窗口的大小可以取的最小值，w表示当前滑动窗口的大小，index表示即将滑出滑动 窗口的记录在数据集中的索引位置，方;代表数据集中索引为i的记录是否与index位置 的记录互为重复，若它们重复，则Bi=1，否则B=0。可以看出，若w内的记录都是 重复记录，则滑动窗口大小更新为最大值wmx，相反，若w内的记录互不重复，则滑 动窗口大小更新为最小值wmn ;并且距离index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index位置的记录互为重复记录时）。

0

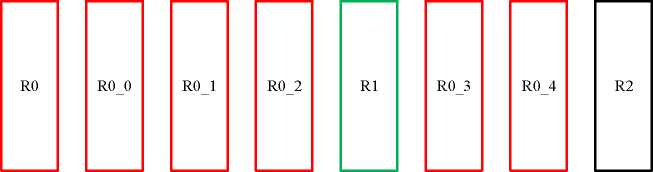
■ w = 4

©

w = 3 •

©

w=4



w = 5



w=4



w=4



图3.3自适应大小的滑动窗口

图3.3展示了自适应大小的滑动窗口检测过程，在图3.3中，R0、R0\_0、R0\_1、 R0 2、R0 3、R0 4互为相似重复记录，记录R0、R1、R2、R3互不重复。滑动窗口

内的归并过程从左向右进行。设定最大窗口为5,最小窗口为3,①表示初始时，滑 动窗口大小为4;由公式(3-2)计算得到第②步所采用的滑动窗口大小变为5,因为此 时窗口内的记录和R0\_2全部重复，所以第②步的窗口大小被扩大；然后由公式(3-2) 计算得到第③步窗口大小取最小值3,因为此时滑动窗口内的数据和R1互不重复， 所以窗口大小被缩小；同理得到第④⑤⑥的滑窗大小。图3.3简明生动地表现了自适 应滑动窗口的变化过程，展现了 OMPN算法所使用的自适应大小的滑动窗口检测方 法的优势。

3.3基于预标记处理排序关键字不完整的方法

1. MPN排序方法的缺陷

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，表3.5给出了一种数据缺失情况的示例。

表3.5缺失数据及不完整数据示例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Record | First Name | Last Name | Address | ID |
| R | Jack | tolfo | 123 First Street | 12345678 |
| R | Jack |  | 123 First Street | 12345673 |
| R | Jack | Stolpho | 123 First Street | 12345678 |
| R | Jacon | Stiles | 123 Forest Street | 12345432 |

在表3.5中，记录R1的Last Name字段原本应该是“Stolfo”，出现字段不完整 而变成“Sto”，在生成排序关键字时若采用3.1.1中的第3种方式，即提取LastName 的前三个辅音字母是则只能得到“s”和“t”两个字母，则排序关键字位数少了一位， 所以生成的关键字为“TLFJCK123FRT213”。记录R2的Last Name字段则直接完全 缺失，所以提取的关键字为“JCK123FRT213”，在排序的过程中，本来属于重复记 录的R1和R2由于排序关键字的首字母的差异而无法聚集在近邻的位置，使得被检 测为互为重复记录的概率减小。

传统的MPN算法在处理这种带有不完整数据和缺失数据的数据集时，就会遇到 这种问题，从而使检测精度降低。为了克服这一缺点，本文提出了针对排序关键字不 完整的改进方法，详细介绍在3.3.2节。

3.3.2改进的基于预标记的方法

缺失数据的处理是数据清洗的另一个分支研宄领域，面对缺失值常见的做法主要

有三种[45]:

1. 以缺失数据的均值、中位数、众数等统计计算结果填充缺失值；
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（2)种做法填充结果可能较为准确但是需要人工干预，工作量较大；第（3) 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，因此，这几种处理 缺失值的方法不能较好的适用于相似重复记录检测问题。

针对相似重复数据检测问题，考虑到不完整数据和缺失数据会造成记录的排序关 键字缺失或不完整，进而会对记录排序后的位置产生影响，所以本文提出了基于标记 的处理数据缺失的方法，该方法的主要操作过程为：

1. 对所有关键字不完整的记录的ID进彳丁标记；
2. 从数据全集中去除第（1)步所标记的数据，只对排序关键字完整的记录进 行排序和归并；
3. 处理被标记的带有缺失值的记录，分别对这些记录进行检测，将其一一聚 类到第（2)步得到的重复数据簇中。

本文这种基于预标记处理缺失值做法能弥补MPN算法在排序关键字缺失的情况 检测效果差的缺点，同时，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，时间耗费在合理的范围内，在真实数据集中，缺失值往往只占有较小的 比例，因此该方法是可行的。

* 1. OMPN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，OMPN算法的改进思想在于以下三点：

1. 基于字段区分度选取排序关键字，避免了对专家经验的依赖性。
2. 采用可伸缩的滑动窗口检测方法，根据数据特点动态调整检测窗的大小， 减少不必要的比较次数。
3. 预标记含有不完整排序关键字的记录，更适用于真实应用场景。

有#趟SNM过程的OMPN算法步骤如算法3.3所示：

算法3.3: OMPN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#组排序关键字;
4. 独立地执行步骤5〜8#次；
5. 按照排序关键字的产生方式对每条记录提取其排序关键字Aey.；
6. 对数据集按照关键字fey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 失关键字记录集合.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dMp\_%(；
8. 将\_此（与此纟\_術+从\_ rncowp/ete \_ 进行重复归并，然后计算此集合的传递闭包

*transitive—closure—seti ；*

1. 将#次SNM重复检测得到的tmns7+t+ve\_c/osure\_set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total \_办户\_set。

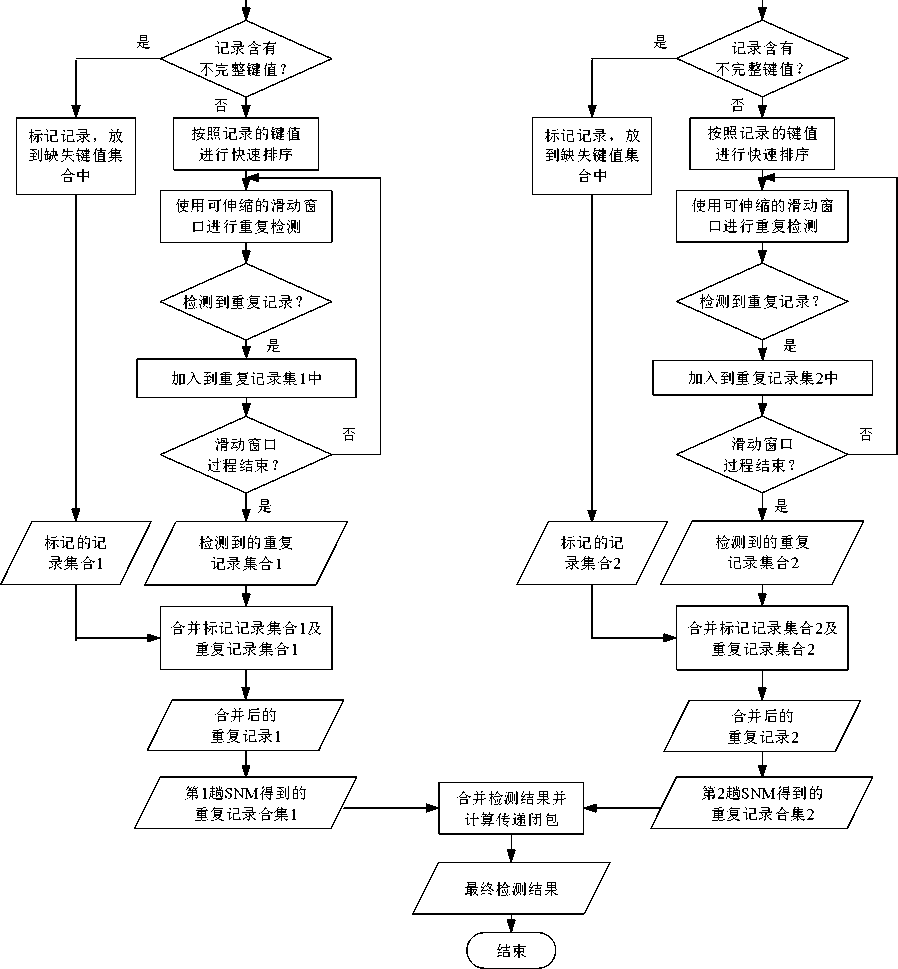
含有两趟SNM过程的OMPN算法流程如图3.4所示:

开始

读取输入数据 计算字段区分度

|  |  |  |
| --- | --- | --- |
| 生成键值方案1 |  | 生成键值方案2 |
| i | | i |
| 生成每条记录 |  | 生成每条记录 |
| 的排序键值 |  | 的排序键值 |

图3.4含有2趟SNM过程的OMPN算法流程图



3.4.2时间复杂度分析

OMPN算法是在MPN算法的基础上进行的改进与创新，因此主要对这两种算法 的时间复杂度进行分析。在相似重复记录检测算法的处理过程中，理想状态下所有的 数据都可以在内存中处理，不考虑磁盘I/O的情况。

MPN算法首先创造排序关键字需要对数据集进行整体遍历，所以该阶段的时间 复杂度为0(#)；排序过程采用快速排序，算法时间复杂度为0(#log#)；滑动窗口 的归并检测过程需要进行^次比较，所以该阶段的时间复杂度为其中W代 表待检测数据集中的记录总数，0代表滑动窗口的大小；在传递闭包的计算过程中， 假设重复记录数据集的大小为^，则该阶段的时间复杂度为〇(，）。所以对于MPN 算法来说，总的时间复杂度为：

Tmpn = 〇( N) + 〇( N log N )+O(^N )+O(d3) (3-5)

由公式(3-3)可以看出，OMPN算法首先需要对数据集中的所有字段进行区分度 统计，假设每条记录的字段总数为々，则区分度统计阶段的时间复杂度为O0N);生 成排序关键字过程、排序过程、以及滑动窗口归并过程与MPN算法的时间复杂度相 同，分别为O(N)、O(NlogN)、O0N)；设因为排序关键字为空而被标注的数据集 包含记录数为m，则与已检测识别出的数据集进行重复记录检测过程的时间复杂度为 O(mN);传递闭包过程中的时间复杂度同理为O(d3)，d为检测结束后重复记录集合 的大小。所以OMPN算法的时间复杂度为（一般情况下带有不完整排序关键字的记 录数m满足m >k并且m >0 ):

T^n = O(kN) + O( N log N)+O(oN)+O(mN)+O(d3) (3-6)

观察公式(3-3)和公式(3-4)组成部分可以发现，两者复杂度在同一数量级，所以两 者的时间复杂度在特定的数据集上是一致的。若用^代表常数，两个公式都可以简化 成下面的式子：

T = O(aN) + O( N log N )+O(d3) (3-7)

当重复记录较多时，即d的值较大，此时O(d3)>O(aN) + O(NlogN)两个算法的 时间复杂度均为O(d3)，这体现出了 MPN算法和OMPN算法的时间消耗受重复数据 比例的影响均较大。

* 1. SNM、MPN、OMPN综合对比实验 3.5.1实验数据介绍

为了方便研宄使用，本文实验采用的数据集是由第三方的数据生成器“febrl” [46] (开源地址：[https://sourceforge.net/projects/febrl/)生成的。“febrl”](https://sourceforge.net/projects/febrl/)%e7%94%9f%e6%88%90%e7%9a%84%e3%80%82%e2%80%9cfebrl%e2%80%9d) 的数据源是澳大 利亚某卫生部门的数据库。生成数据集中记录包含的字段及字段含义如下表所示：

表3.6记录字段说明

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 字段描述 | 举例1 | 举例2 |
| rec\_id | 记录ID | rec-454-org | rec-454-dup-0 |
| culture | 文化 | pak | pak |
| sex | 性别 | f | f |
| age | 年龄 | 30 | 30 |
| date\_of\_birth | 出生日期 | 19870221 | 19870221 |
| title | 头衔 | hon | hon |
| given\_name | 名字 | sophie | sophie |
| surname | 姓氏 | bozdar | bozdaa |
| state | 州 |  |  |
| suburb | 郊区 | holsworthy | holsworthy |
| postcode | 邮编 |  |  |
| street\_number | 街道号码 | 46 | 46 |
| address」 | 地址1 | thurgood court | thurgood court |
| address\_2 | 地址2 |  |  |
| phone\_number | 电话号码 | 08 42167414 | 08 42167414 |
| soc\_sec\_id | 社保ID | 3920942 |  |

采用数据生成器的好处在于：生成器公开的接口中提供了多个参数，这些参数能 够方便用户自定义数据集的大小、重复比例、字段特征、错误类型、重复记录的概率 分布等，由该生成器得到的数据集非常接近现实数据；相较于真实数据，生成的数据 记录拥有唯一的标识符，更方便后期对算法的查准率、查全率等进行计算和评估。 “febrl”的公开参数列表及其说明如下表所示：

表3.7 febrl公开接口的参数说明 参数名称 参数说明

|  |  |
| --- | --- |
| outputFileName | 输出文件名（.CSV格式） |
| numberOfOriginalRecords | 原始数据集大小 |
| numberOfDupRecords | 由原始数据集生成的重复数据集大小 |
| maxNumOfDupPerRec | 一条原始记录能够最多生成的重复记录个数 |
| maxNumOfModPerRec | 一个字段最多可修改数目 |
| maxNumOfModPerRec | 一条记录最多可修改字段数目 |
| probabilityOfDup | 重复记录在数据集中的概率分布  (均匀分布、泊松分布或齐夫分布） |
| typeOfModification | 字段可能发生的错误类型（typo:印刷错误、ocr:  扫描错误、phonetic:发音错误或者以上所有） |

3.5.2算法的评价指标

衡量相似重复记录检测算法效果的常用标准主要有查全率（reca//)、查准率 (precision)以及 F-measure[13K14]〇

将算法的重复记录检测结果与实际数据集重复记录进行比较时，会出现以下四种 可能的情况：

1. TruePositive (ZP):算法判定为重复记录，实际上也是重复记录；
2. False Positive (FP):算法判定为重复记录，但实际不是重复记录；
3. True Negative (Z5V):算法判定为非重复记录，实际也不是重复记录；
4. False Negative (F#):算法判定为非重复记录，但实际上却是重复记录。 查全率（reca//)代表了算法检测重复记录是否完备的能力，它的计算方法是算

法检测出的正确重复数除以实际数据集中的重复记录总数，如公式(3-6)所示：

*reca// =*

*TP*

*TP + FN*

(3-8)

查准率（precision)代表了算法正确识别重复记录的能力，它的计算方法是算法 检测出的正确重复数除以被算法识别为重复记录的总数，如公式(3-7)所示：

*TP*

(3-9)

*precision =*

*TP + FP*

F-measure是查全率和查准率的加权调和平均，F-measure的计算公式如(3-8):

*^ \_ (a2 +*1)*precision* x *recall a a precision + recall*

其中参数a取值为1时，F-weaswre即为最常见的^FV-weaswre:

(3-10)

^ 2 *precision x recall*

(3-11)

*Fx =—*

*reecision + recall*

假设有8条记录K、八、怂、&、足、及、矣、&，其中{及，八，怂}和 {&，尾，及}互为相似重复记录，若通过算法检测出的重复结果为{及，八，怂， &}和{&，尽，及，凡}互为相似重复记录，则TP=4, FP=2，TW=2, FW=0,所以 算法的查全率为 4/(4+0)=100.00%， 查准率为 4/(4+2)=66.67%， F7-TOeaswre=2\*1\*0.6667/(1+0.6667)=80.00%。

3.5.3实验结果分析

在不同规模的数据集上进行实验，对OMPN算法、MPN算法和SNM算法的查 全率和查准率进行综合对比。这三种算法的实验参数为：

OMPN算法的实验参数：最大滑动窗口最大为0max =20，最小为0mn=3，进行单 趟SNM的次数/=3；

MPN算法的实验参数：固定滑动窗口大小为0=5，进行单趟SNM的次数/=3；

SNM算法的实验参数：固定滑动窗口大小为0=5。

在字段相似度检测过程中用到的两个常数VERY\_CLOSE\_CONSTANT (字符串 非常接近）、CLOSE\_CONSTANT (字符串比较接近），对它们分别赋值： VERY—CLOSE—CONSTANT=0.8，CLOSE—CONSTANT=0.6。

数据集的规模与生成数据的参数取值如表3.8所示：

表3.8测试数据集参数取值说明

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 数据集 | P1 | Pi | P3 | P4 | P5 | P6 | P7 |
| dataset1 | 5000 | 1000 | 3 | 1 | 1 | uniform | phonetic |
| dataset2 | 10000 | 2000 | 3 | 1 | 1 | uniform | phonetic |
| dataset3 | 20000 | 4000 | 3 | 1 | 1 | uniform | phonetic |
| dataset4 | 50000 | 10000 | 3 | 1 | 1 | uniform | phonetic |
| dataset5 | 80000 | 16000 | 3 | 1 | 1 | uniform | phonetic |
| dataset6 | 100000 | 20000 | 3 | 1 | 1 | uniform | phonetic |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| dataset7 | 200000 | 40000 | 3 | 1 | 1 | uniform | phonetic |
| dataset8 | 500000 | 100000 | 3 | 1 | 1 | uniform | phonetic |

表3.8中的参数开〜外代表的含义分别为：A-原始记录数、p2-重复记录数、A -单个记录最多重复数、p4-单个字段最多修改数、ft-单个记录最多修改字段数、p6 -重复记录的概率分布、p7错误类型。

本实验的实验环境配置如表3.9所示：

表3.9实验环境配置

|  |  |
| --- | --- |
| 操作系统 | Windows 7旗舰版 |
| 处理器 | Core i7 (8 核心） |
| 内存大小 | 8G |
| JDK版本 | JDK1.8.0 |
| JVM配置 | -Xmx4096m (堆内存最大4G) |

在表3.8所示的8种不同规模的据集上，分别使用SNM、MPN和OMPN进行实 验，统计每种算法在每种数据集上的reca//、和运行时间（单位：秒），实 验结果分别如表3.10至表3.12所示。

表3.10 SNM、MPN、OMPN算法查全率对比表

|  |  |  |  |
| --- | --- | --- | --- |
| 数据集大小（千条） | SNM 算法（°%) | MPN算法（％) | OMPN 算法（％) |
| 5 | 80.60 | 94.80 | 98.80 |
| 10 | 78.15 | 91.80 | 98.40 |
| 20 | 78.03 | 89.28 | 98.75 |
| 50 | 78.01 | 87.39 | 97.90 |
| 80 | 76.38 | 82.08 | 98.58 |
| 100 | 75.85 | 80.76 | 98.73 |
| 200 | 73.83 | 79.72 | 99.10 |
| 500 | 72.90 | 78.91 | 96.91 |

表3.11 SNM、MPN、OMPN算法查准率对比表

|  |  |  |  |
| --- | --- | --- | --- |
| 数据集大小（千条） | SNM算法（％) | MPN算法（％) | OMPN 算法（％) |
| 5 | 100.00 | 99.89 | 99.50 |
| 10 | 99.94 | 100.00 | 99.54 |

|  |  |  |  |
| --- | --- | --- | --- |
| 20 | 99.97 | 100.00 | 98.78 |
| 50 | 99.87 | 99.98 | 96.93 |
| 80 | 99.81 | 99.92 | 95.69 |
| 100 | 99.92 | 99.87 | 94.77 |
| 200 | 99.73 | 99.89 | 89.70 |
| 500 | 99.53 | 99.70 | 88.17 |

|  |  |  |  |
| --- | --- | --- | --- |
| 数据集大小（千条） | SNM算法（s) | MPN算法（s) | OMPN 算法（s) |
| 5 | 0.503 | 0.938 | 1.274 |
| 10 | 1.196 | 2.320 | 2.961 |
| 20 | 3.540 | 6.985 | 10.327 |
| 50 | 17.169 | 26.347 | 59.420 |
| 80 | 47.255 | 239.434 | 197.439 |
| 100 | 67.173 | 372.245 | 290.903 |
| 200 | 237.805 | 1668.995 | 1391.415 |
| 500 | 1493.305 | 10464.894 | 11296.066 |

表 3.12 SNM、MPN、OMPN 算法运行E

间对比表

根据表3.10-表3.12的实验结果，分别作出查准率、查 如图3.5至图3.7所示。

全率和运行时间的折线图，

—SNM V—MPN ■e—IMPN

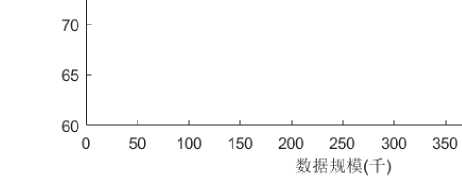
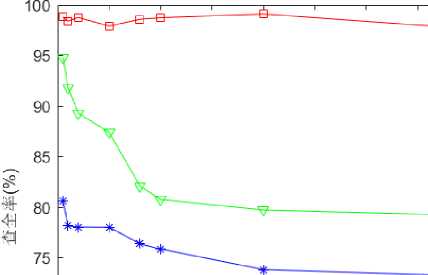
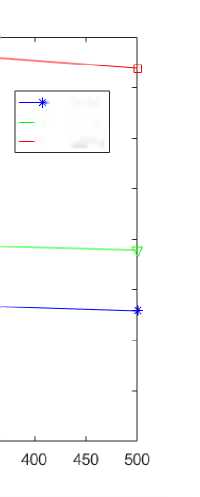
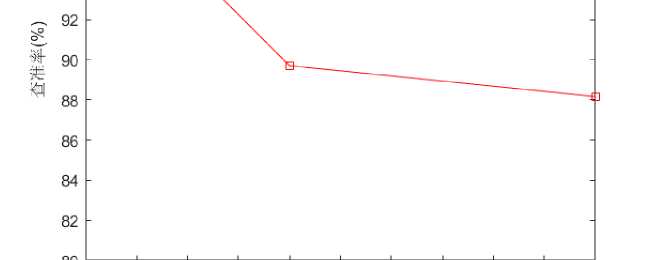
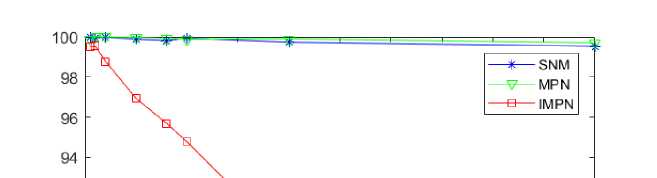


图3.5 SNM、MPN、OMPN算法查全



率折线图

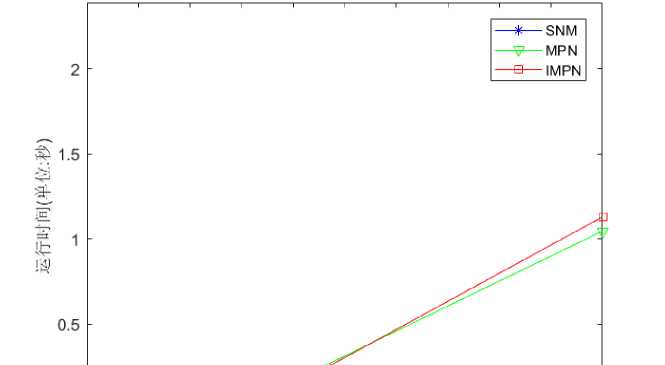


0 50 100 150 200 250 300 350 400 450 500

数据规模(千J

图3.6 SNM、MPN、OMPN算法查准率折线图

K104



0 50 100 150 200 250 300 350 400 450 500

数据规模(千）

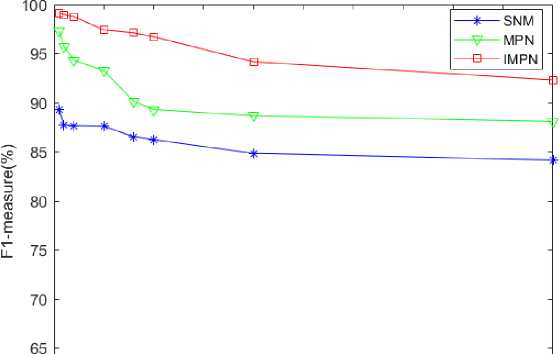
图3.7SNM、MPN、OMPN算法运行时间折线图

根据表3.10至表3.12计算得到三种算法的每种数据集上的:F1值，如表3.13所 示，并作出如图3.8所示的F1值折线图。

表 3.13 SNM、MPN、OMPN 算法 F1-measure 值对比表

|  |  |  |  |
| --- | --- | --- | --- |
| 数据集大小（千条） | SNM算法（％) | MPN算法（％) | OMPN 算法（％) |
| 5 | 89.258 | 97.278 | 99.149 |
| 10 | 87.712 | 95.725 | 98.967 |

|  |  |  |  |
| --- | --- | --- | --- |
| 20 | 87.648 | 94.336 | 98.765 |
| 50 | 87.597 | 93.262 | 97.413 |
| 80 | 86.537 | 90.126 | 97.114 |
| 100 | 86.237 | 89.304 | 96.709 |
| 200 | 84.847 | 88.672 | 94.166 |
| 500 | 84.159 | 88.095 | 92.334 |



0Q | | | | | | | | |

0 50 100 150 200 250 300 350 400 450 500

数据规模(T\_)

图 3.8 SNM、MPN、OMPN 算法 F1-measure 折线图

由图3.5可以看出：改进的OMPN算法面对不同大小的数据集（其他条件如滑 动窗口大小、数据集特征参数等均一致)，查全率均高于SNM算法和MPN算法，并 且OMPN算法查全率的数值稳定在96%以上。所以证明了改进的OMPN算法在查全 率上相较于传统的MPN算法拥有较大的提升；随着数据集的增大，SNM算法和MPN 算法的查全率均呈下降趋势，而OMPN算法的查全率较为稳定。

由图3.6可以看出：面对不同的数据集，SNM和MPN算法表现较好，这是因为 算法中采用的相似记录判等方法是基于当前数据集特征的判定规则，所以需要领域专 家的经验和知识；OMPN算法采用基于字段区分度的加权判等方式，在检测大小从 5000到200000的数据集时，查准率均在90%以上，属于尚可接受的范围，但是当数 据集大小达到500000时，查准率低于90%，所以OMPN算法的查准率还有待改进的 空间。本文的第四章针对这一问题提出了新的改进思路，并取得了良好的效果。

由图3.7可以看出：SNM算法时间消耗最低，因为MPN和OMPN算法均包含 多趟独立的SNM过程，并且含有传递闭包的计算；与MPN相比较，OMPN算法采 用了自适应大小的滑动窗口，这一改进是可以减少时间消耗的，但是OMPN算法还 包括对含有空（或者不完整）排序键值的记录进行归并的过程，所以总体的时间和 MPN算法差距不大。

综合以上结果可以看出，OMPN算法相对于MPN算法较为明显的改进在于：在 和MPN消耗时间差距非常小的同时实现更高的查全率；算法的缺点也很明显，即随 着数据量的增大，相似重复记录判等方法表现较差，导致查准率的下降。但是由图 3.8可以看出，综合考虑查全率和查准率，OMPN算法的整体表现要好于MPN。

3.6本章小结

本章首先介绍了 MPN算法的缺点，然后在此基础上提出了一种改进的算法 OMPN，并从三个方面介绍了传统MPN的处理方法的不足以及OMPN算法的对其缺 点的改进。之后介绍了 OMPN算法的设计与流程步骤，最后通过采用数据生成器febrl 生成的不同规模的数据集进行对比实验，实验验证了 OMPN算法的查全率和F1值都 较高，比传统的SNM算法和MPN算法更优，但是也反映了 OMPN算法查准率不够 理想的问题，因此，本文继续对OMPN算法进行改进，详细介绍在第四章。

第四章基于遗传神经网络改进的OMPN算法

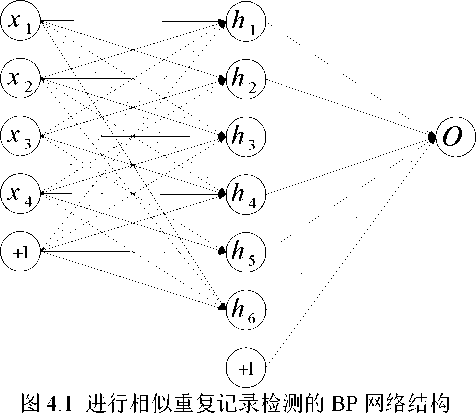
OMPN算法采用“排序/归并”的思想，在归并过程中，需要采用有效的方式对 滑动窗口内的记录进行判等，OMPN采用基于字段区分度的加权判等方式，对不同 的字段分配相应的权值，这种方式表现不稳定，随着数据量的增加算法的查准率下降。 传统的MPN算法主要根据人工经验选择有效的字段对不同的记录进行判断，因此， 受到人为因素的影响较大。为了避免这个弊端，同时提高OMPN算法的查准率，本 章提出了基于遗传算法和神经网络相结合的、有监督地学习判等方法，并与OMPN 的算法思想相结合，提出增强的MPN算法（Advanced - Optimized Multi-Pass Sorted Neighborhood，A-OMPN)。A-OMPN算法的查全率和查准率都优于现有算法，但是 遗传神经网络的训练耗时较大，所以，我们在A-OMPN的基础上进行简化，只训练 BP神经网络进行判等操作，由此得到基于BP网络的MPN算法（BP-based Optimized Multi-Pass Sorted Neighborhood, BP-OMPN) 〇

本章首先介绍了遗传神经网络在相似重复记录检测问题上的操作过程，然后分别 给出A-OMPN算法和BP-OMPN算法的具体操作过程，并通过实验验证了本文提出 的A-OMPN算法和BP-OMPN算法的性能。

4.1遗传神经网络用于相似重复记录检测

1. BP神经网络的设计

BP神经网络是一种前馈传播神经网络，因采用BP (Back Propagation)算法进行 网络模型的训练而得名。



BP神经网络一般由3层或者3层以上神经元组成。一方面，由引理[47]看出：三 层的反向传播神经网络能够对非线性函数实现任意精度的逼近[47]。另一方面，网络层 数的增加虽然可以增强表达能力，但是也会带来较高的训练复杂度。因此，针对相似 重复记录检测问题，采用简单的三层BP神经网络，图4.1表示了用于相似重复记录 检测的BP网络结构，网络各层所代表的含义及其计算方法分别为：

1. 输入层的结点数由数据集中记录的字段数目确定，输入层的取值是两条记 录的相似度向量。若数据集中的记录含有m个字段，则输入层结点数为rn。任取两 条记录，计算这两个记录在第/个字段下的相似度值为《+%，/=1，2，...m，则这两条记 录的相似度向量为，BP网络的输入层各结点的取值也随之确定， 即第+个结点的输入为《+m+。例如，对于表3.6所示的字段说明数据，每条记录都有 16个字段，其中，第一个字段“rec」d”是为了方便计算算法的查重效果而设计的， 不属于数据集的特征，所以这个数据集共有15个字段，因此，输入层应当有15个结 点（不含一个偏置结点）。
2. 输出层的结果代表BP网络对于两条记录相似度的计算。结点个数为1，输 出的结果介于0到1之间，结果越接近1表示两条记录相似程度越高。
3. 隐含层连接输入层与输出层，主要进行BP网络的前向计算和误差反向传 播更新权值的过程。结点数目由公式(4-1)[48]计算得到。

Ml =^l 0.43NK + 0.12K2 + 2.54 N + 0.77 K + 0.35 +0.51

(4-1)

J N + 0.618(N -K),N > K [N-0.618(K -N),N < K



M = max(M”M2)

其中M代表隐含层结点的个数，W代表输入层结点的个数，K代表输出层结点 的个数，计算过程的结果需要四舍五入进行修整。例如，使用表3.6所示的字段说明 表进行BP网络结构的设计时，输出数值个数为1，所以K=1; 15个字段对应着输入 神经元的个数为W=15,通过公式(4-1)计算得到叫=7，M2=24,最终得到适用于该 数据集的BP网络的隐层神经元个数为M=24。

本文使用的激活函数为Sigmoid函数[39,40,41]，函数表达形式如公式(2-9)所示。

神经网络的初始权值对网络最终的收敛情况具有重要的作用，同时，初始权值也 会影响训练速度的快慢[49]。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sigmoid激活函数的导数取最大值的地方进行下一轮权值的调 节与变化，在本文中，初始权值取(-1,1)之间的随机数。

权值更新的学习速率决定了每次训练结束后权值更新的幅度，过大的学习速率会

导致系统震荡，过小的学习速率又会使得权值在每一次更新过程中的改变量较小，进 而减慢训练过程，耗时较大[50]。通常，可以采用小学习速率，以保证系统稳定性而不 会导致修改幅度过大，本文取值为0.15。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。

BP神经网络训练终止的条件是训练误差达到预先设定的阈值或权值更新次数达 到预先设定的迭代次数，本文对神经网络的实现过程中，同时采用这两种终止条件， 只要满足其中一条，则训练终止。

针对相似重复检测问题，将已知记录是否重复的数据作为训练集，用于训练神经 网络，得到训练完成的网络后，用该网络对测试数据进行相似重复检测。使用BP网 络进行相似重复记录检测的算法操作过程如图4.2所示：

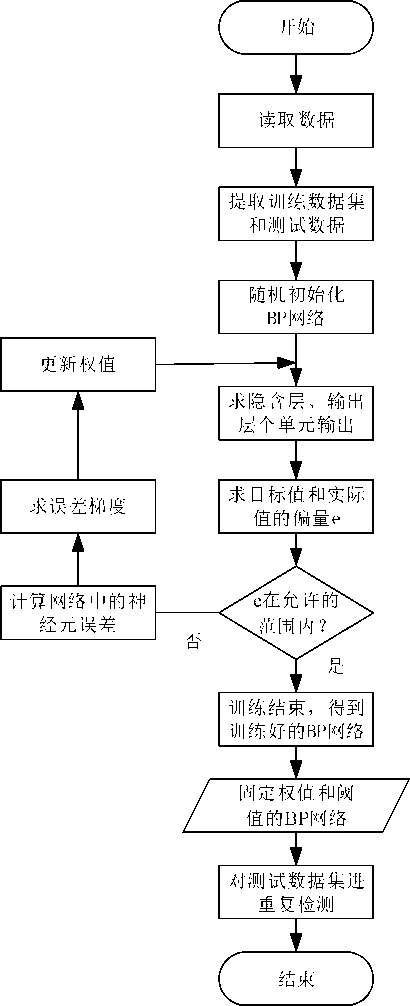


图4.2 BP网络检测重复记录的流程图

4.1.2基于遗传算法改进的神经网络

BP神经网络通过训练能够有效地对测试数据集中的相似重复记录进行检测判断， 解决了大数据量情况下传统的基于“排序/归并”思想的检测算法检测效果较差的问 题，并且拥有较好的适应性。但训练好的BP神经网络可能达到局部最优状态。GA (Genetic Algorithm)算法[51]是一种模拟生物进化过程的进化算法，它利用自然界的 进化思想，通过选择、交叉、变异等操作实现种群的进化过程，并在演化过程中淘汰 适应性较差的个体，经过数代进化之后筛选出适应度较高的个体，得到全局的最优解

[51]。遗传算法拥有较强的全局搜索能力，所以可以将其应用到BP神经网络中，以解 决神经网络难以跳出局部最优的问题[52][53]。

基于遗传神经网络求解相似重复记录检测的遗传算法框架的元素主要有：

1. 染色体

采用实数编码[54]的编码方式，将BP神经网络的权值和阈值作为一个独立的染色 体，设种群规模为户，则共有户组神经网络权值和阈值，即需要训练户个BP神经网 络。

以图4.1所示的神经网络为例，将训练得到的BP神经网络的权值以及阈值转化 为染色体，输入层包含4个神经元，1个偏置单元，隐含层包含6个神经元，所以输 入层到隐含层之间的权值共有(4+1)\*6=30个，输出层有1个神经元，所以隐含层到输 出层之间共有6+1=7个权值，因此，该神经网络共有37个权值，1个输出层阈值， 当采用实数编码方式时，染色体的长度为38。

1. 适应度函数

在重复检测问题中，适应度函数用于衡量相似重复检测结果和真实结果的之间的 差距，差距越小，说明这个染色体所代表的的BP神经网络对数据的检测结果越准确， 所以适应度值越大，相反，染色体的适应度就越小。本文使用公式(4-2)[55]计算第z个 个体的适应度值。

1

(4-2)

1+E

公式(4-2)中，E.代表使用第z个个体对应的BP神经网络进行重复记录检测时， 在记录数据集上所得到的总误差，E.的计算公式如公式(4-3)所示。

E,=：£(g-a)2 (4-3)

*j=1*

公式(4-3)中，#代表训练数据集的记录总数，g代表第j条记录的期望输出结 果，flj代表第j+条记录的实际输出结果。

1. 选择

采用轮盘赌[56]方法进行染色体的选择，使得适应度函数值较大的个体被选中的概 率较大。轮盘赌算法的计算如公式(4-4)所示：

M

Z fi

(4-4)

i=1

其中，M代表种群中个体的数量，fi代表第i个染色体的适应度函数的值。

1. 交叉、变异[44]

因为一个网络权值向量构成一个染色体，所以在该问题上，采用实数编码，因此 对任意不同的两条染色体上的对应位置进行数值交换，即完成一次交叉操作，本文采 用两点交叉操作。

变异算子模仿的是基因突变的过程：染色体某个位置上的基因突变成为其等位基 因，从而可能引发性状表现上的变异。

使用遗传神经网络进行相似重复记录检测的执行过程图如图4.3所示。其主要操 作为：首先，进行有监督的训练学习得到户组：8?网络；然后，用这户组：8?网络的 权值产生初始状态下的种群，种群中含有户条染色体，种群经过遗传操作不断得到优 化；最后，从进化结束后的种群中获取最优的BP网络，使用这组最优的网络权值所 代表的网络对待检测的数据集进行重复性检测。

使用遗传神经网络方法进行相似重复记录检测时，需要对数据集中的任意两个不 同的记录进行比较，判断他们是否相似或重复，因此时间复杂度较高，为〇(#2)，存 在冗余判断过程。所以需要对其进行改进，在保留遗传神经网络所具有的优势的同时， 减小算法的时间复杂度。

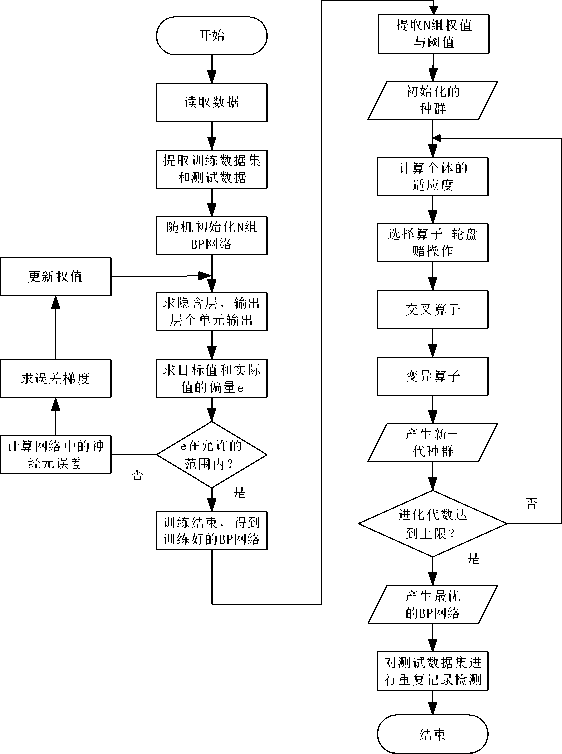


图4.3遗传神经网络检测相似重复记录流程图

4.2基于神经网络的A-OMPN算法和BP-OMPN算法

A-OMPN算法和BP-OMPN算法的操作过程都包括两部分：（1)根据OMPN算 法对原数据进行处理，包括基于字段区分度提取排序关键字、依据关键字对所有记录 进行排序、选取自适应大小的滑动窗口；（2)使用某种判等方法，对滑动窗口内的数 据进行判断，最终得到重复记录。与传统的基于遗传神经网络求解相似重复记录问题 相比，A-OMPN算法和BP-OMPN算法首先通过OMPN算法对数据集进行预处理， 这样就避免了大量冗余的判等操作，因此时间复杂度降低。A-OMPN算法使用遗传 神经网络进行判等，BP-OMPN算法仅使用BP神经网络进行判等。

4.2.1基于遗传神经网络的A-OMPN算法

传统SNM算法和MPN算法使用的是基于专家经验知识的规则产生式系统 (OPS5)，这种判等方法主要依赖人工操作以选择进行比较的字段。第三章提出了基 于字段区分度的加权判等方式，对不同的字段分配相应的权值，但是这种方法对小数

据量的数据表现较好，当数据规模不断增大时，算法的查准率逐渐变差，性能不稳定。 传统的基于遗传神经网络进行相似重度检测的算法不存在排序操作，所以需要在数据 全集对数据进行两两判断，时间复杂度为〇(#2)。

本文提出了 A-OMPN算法，经过训练完成的神经网络判断滑动窗口内的记录是 否重复，这样可以根据数据集的特点进行动态调整，减少了人工干预，同时可以提高 算法对多种数据集的适应性。

算法对于给定的训练集，共训练册\_个BP神经网络，然后用训练得到的^\_个 BP神经网络的权值作为初始种群，再经遗传算法进行操作，不断优化，最后从^\_个 BP神经网络中选择最优的结果作为最终训练得到的神经网络，然后用这个神经网络 进行判等操作，A-OMPN算法的操作过程如算法4.1所示：

算法4.1: A-OMPN算法

1. 读取待检测的数据集以及训练数据集，设定执行SNM的次数为^^ ；
2. 随机产生个初始BP神经网络；
3. 分别对个BP网络进行训练，用训练完成的个BP网络的权值向量产生初始种群；
4. 种群进行遗传操作直到达到最大进化次数，得到最优的BP网络权值，即最优网络5/^,；
5. 根据字段区分度选取排序关键字；
6. 对数据集中的每一条记录：

如果该记录的关键字存在缺失：

将该记录的ID加入到缺失关键字记录集合中；

1. 对数据全集中的、非中的其余记录进行排序；
2. 采用5/^,进行可伸缩大小的滑动窗口重复检测，得到重复集合D2 ；
3. 在集合D2中检测Dj中的记录的重复记录，得到最终的重复记录集合D ；
4. 独立地执行步骤5〜9 ,\_次，得到,\_个重复记录集合，并对个重复记录集合进行传递 闭包计算，得到最终重复记录集合Df ；

使用A-OMPN算法进行相似重复记录检测的流程图如图4.4所示。

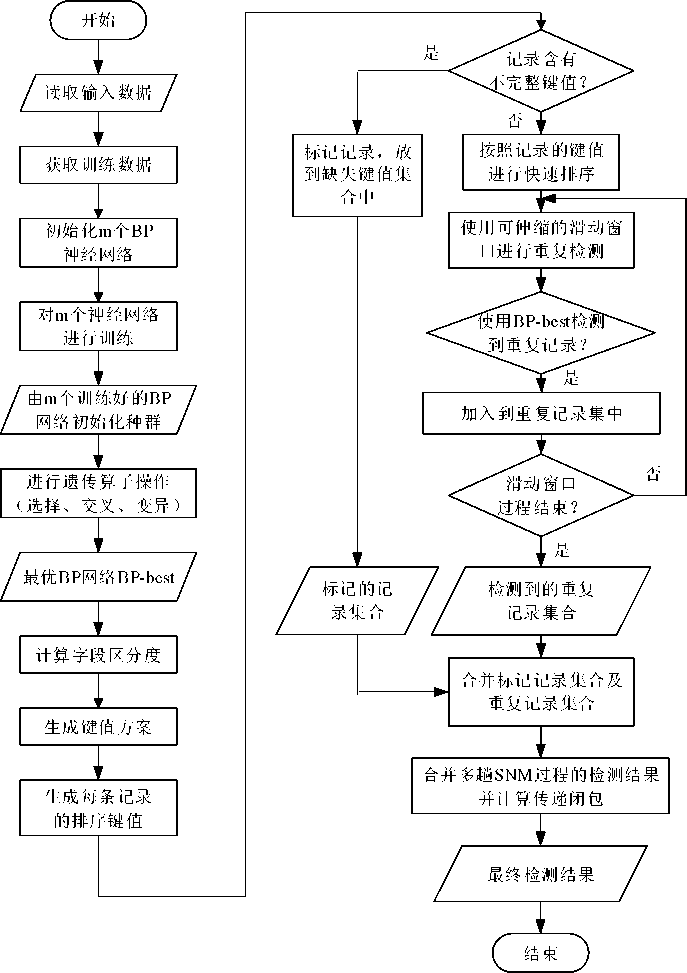


图4.4 A-OMPN算法流程图

与OMPN算法的基于字段区分度的加权判等操作相比，使用遗传神经网络进行 判等可以有效地提高算法的查准率。同时，与传统的使用遗传神经网络求解该问题的 算法相比，OMPN算法对数据集进行了预处理，只需要比较滑动窗口范围内的记录， 减少了不必要的判断计算，时间复杂度降低。

4.2.2基于BP网络的BP-OMPN算法

A-OMPN算法的需要较长的时间进行网络的训练，因此对该算法进行简化，提 出基于BP神经网络的BP-OMPN算法。设BP神经网络的输入层节点个数为巧，隐 含层节点个数为％，则一次训练过程（包括前向计算输出和误差反向更新过程）的 时间复杂度为设BP网络终止训练过程的最大次数为rSP，遗传算法的种群数量为《\_，则需要训练^\_个BP网络以产生初始种群，所以种群初始化过程的时 间复杂度为种群演化过程的时间复杂度为〇如\_勹，所以A-OMPN算 法训练过程的总时间复杂度如公式(4-8)：

(4-5)

TA—IMPN train

*0(m*size*n*1*n*H ), *n*i*n*H ^ *m*s,ze

*n*i*n*H <*m服*



可以看出算法训练过程的时间复杂度为平方级别。除此之外，BP神经网络优化 算法属于高维优化问题，算法训练的结果往往是收敛于某个鞍点附近，而不是局部最 小值[57]。在鞍点和局部极小值的梯度都等于零，大量鞍点的存在才是神经网络优化困 难的真正原因，而基于遗传算法改进的BP神经网络算法对于拥有较多局部极值的问 题效果较好，适用较为局限，并且算法的搜索效率低下[58]。因此，本节提出了简化的 BP-OMPN算法，仅训练单一的BP神经网络，代替A-OMPN算法中遗传伸进网络的 训练。BP-OMPN步骤如算法4.2所示：

算法4.2: BP-OMPN算法

1. 读取待检测的数据集以及训练数据集，设定执行SNM的次数为〖\_ ；
2. 使用训练数据集训练出一个BP神经网络；
3. 根据字段区分度选取排序关键字；
4. 对数据集中的每一条记录：

如果该记录的关键字存在缺失：

将该记录的ID加入到缺失关键字记录集合中；

1. 对数据全集中的、非中的其余记录进行排序；
2. 采用进行可伸缩大小的滑动窗口重复检测，得到重复集合D2 ；
3. 在集合D2中检测中的记录的重复记录，得到最终的重复记录集合D ；
4. 独立地执行步骤5~9次，得到个重复记录集合，并对个重复记录集合进行传递闭 包计算，得到最终重复记录集合Df ；

BP-OMPN算法流程图如图4.5所示：

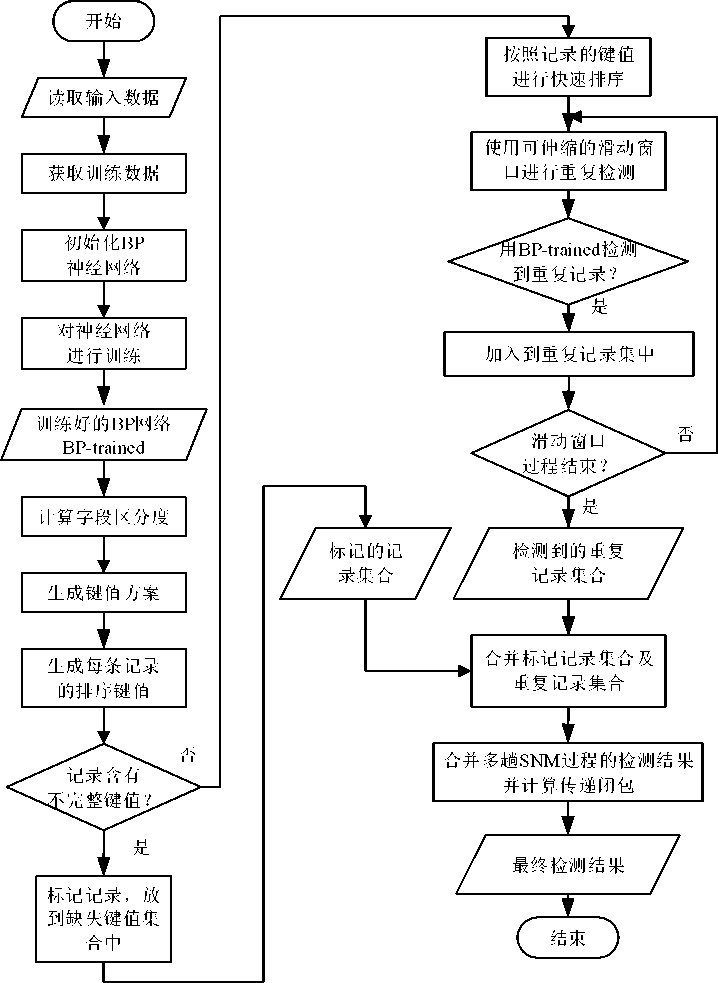


图4.5 BP-OMPN算法流程图

4.3 OMPN、A-OMPN、BP-OMPN 综合对比实验

本节对OMPN算法、A-OMPN算法和BP-OMPN算法在不同数据集上进行实验， 并统计实验结果。

实验环境设置和数据集设定与第三章3.5节相同，均采用8个规模不同但生成方 式相同（生成器“febrl”的参数一致）的数据集，对于神经网络的训练，训练过程采 用的数据集大小为500,即含有500条记录，使用“febrl”生成，生成参数与实验数 据集一致，详情见表3.7与表3.8。

对于OMPN算法，最大滑动窗口大小0mx=2〇，最小滑动窗口大小0mn=3，算法 过程中进行单趟SNM的次数=3。

对于BP神经网络的训练，学习速率设置为0.05,动量系数设置为0.9[59]，网络 最大训练次数rBP=5000；遗传算法种群初始大小WMZ6=50，变异概率0\_&=〇.〇5，种 群最多迭代=300次。

OMPN算法、A-OMPN算法、BP-OMPN算法的查全率、查准率、运行时间（单 位：秒）的实验结果分别如表4.1至表4.3所示，对于A-OMPN算法和BP-OMPN算 法，这里统计的运行时间是使用训练好的网络进行检测的时间，不包括训练网络的时 间。

表4.1 OMPN、A-OMPN和BP-OMPN算法查全率对比表

|  |  |  |  |
| --- | --- | --- | --- |
| 数据集大小（千条） | OMPN 算法（％) | A-OMPN 算法（％) | BP-OMPN 算法（％) |
| 5 | 98.80 | 98.90 | 98.80 |
| 10 | 98.40 | 98.35 | 98.15 |
| 20 | 98.75 | 98.90 | 98.90 |
| 50 | 97.90 | 97.85 | 97.65 |
| 80 | 98.58 | 98.53 | 98.39 |
| 100 | 98.73 | 98.76 | 98.78 |
| 200 | 99.10 | 99.15 | 99.50 |
| 500 | 96.91 | 97.23 | 97.43 |

表4.2 OMPN、A-OMPN和BP-OMPN算法查准率对比表

|  |  |  |  |
| --- | --- | --- | --- |
| 数据集大小（千条） | OMPN 算法（％) | A-OMPN 算法（％) | BP-OMPN 算法（％) |
| 5 | 99.50 | 100.00 | 100.00 |
| 10 | 99.54 | 100.00 | 100.00 |
| 20 | 98.78 | 100.00 | 99.98 |
| 50 | 96.93 | 99.87 | 98.95 |
| 80 | 95.69 | 99.85 | 98.80 |
| 100 | 94.77 | 99.88 | 98.81 |
| 200 | 89.70 | 99.68 | 97.81 |
| 500 | 88.17 | 99.21 | 97.69 |

表4.3 OMPN、A-OMPN和BP-OMPN算法运行时间对比表

|  |  |  |  |
| --- | --- | --- | --- |
| 数据集大小（千条） | OMPN 算法（s) | A-OMPN 算法（s) | BP-OMPN 算法（s) |
| 5 | 1.274 | 21.110 | 20.800 |
| 10 | 2.961 | 83.169 | 77.989 |
| 20 | 10.327 | 336.404 | 328.788 |
| 50 | 59.420 | 2172.766 | 1928.253 |
| 80 | 197.439 | 5510.201 | 5287.167 |
| 100 | 290.903 | 8505.129 | 8119.437 |
| 200 | 1391.415 | 32842.004 | 30901.765 |
| 500 | 11296.066 | 204281.991 | 193198.802 |

根据表4.1至表4.3所示的查全率、查准率和运行时间（单位：秒），分别作出 OMPN算法、A-OMPN算法和BP-OMPN算法的查全率折线图、查准率折线图和运 行时间折线图，如图4.6至4.8所示。

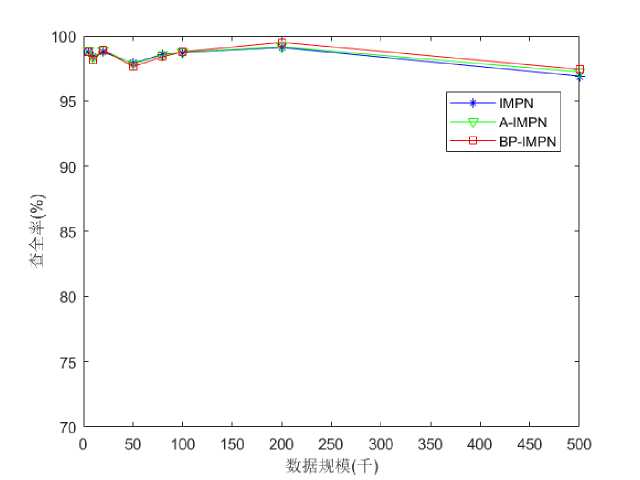


图4.6 OMPN、A-OMPN、BP-OMPN算法查全率折线图

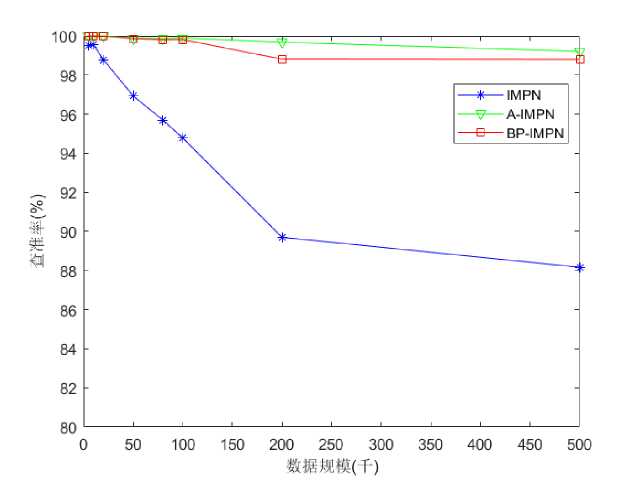
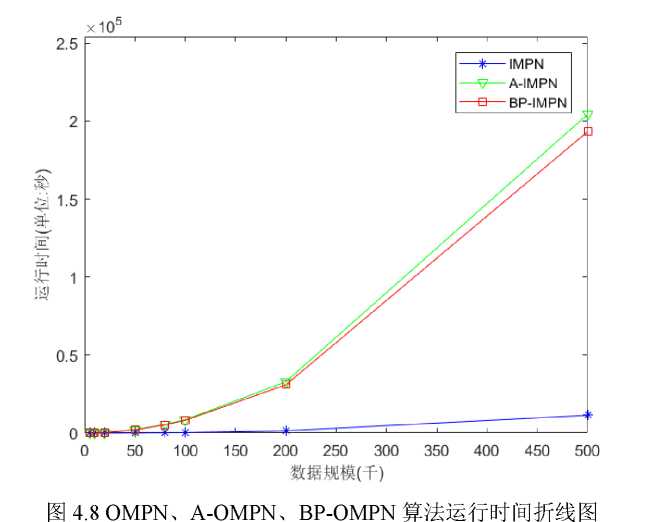


图4.7 OMPN、A-OMPN、BP-OMPN算法查准率折线图



由图4.6所示的查全率对比结果可以看出，A-OMPN算法、BP-OMPN算法与 OMPN算法相比，查全率非常接近，且均在96%以上，证明基于OMPN改进的 A-OMPN算法和BP-OMPN算法能够保证较好的查全率，这与理论上的结果一致， 因为基于神经网络改进的算法和OMPN算法的主要区别在于判等过程采用了不同的 方法，而这一区别几乎不影响查全率的指标。

由图4.7所示的查准率对比结果可以看出，A-OMPN算法和BP-OMPN算法相对 于OMPN算法，查准率均有较大的提升；由图4.8所示的运行时间对比结果可以看

出，A-OMPN算法和BP-OMPN算法的检测时间相近，且数倍于OMPN算法。取得 此实验结果的理论原因在于：两者最主要的区别即是相似重复记录判等阶段，改进的 算法采用遗传算法改进BP神经网络，然后通过学习训练数据集字段相似与整体记录 相似的非线性关系，可以较为准确地对两条记录相似与否进行预测和判断，但同时神 经网络前向传播的计算时间比OMPN算法采用基于权重的判等过程时间复杂度高， OMPN算法的时间复杂度是常数级的0(1),而改进的A-OMPN算法和BP-OMPN算 法使用BP神经网络进行判等，对于~个输入结点、~个隐含结点的三层网络，其 进行检测的时间复杂度为0(Wff),因此，A-OMPN算法和BP-OMPN算法的检测时 间较长。

接下来对A-OMPN算法和BP-OMPN算法在多个不同数据集上进行训练，并统 计训练时间，分析这两种算法的训练效率。

实验采用5个规模不同但生成方式相同（生成器“febrl”的参数一致）的数据集， 训练数据集的规模与生成数据的参数取值如表4.4所示。数据集的大小分别为100、 200、500、1000、2000，重复记录所占的比例均为20%。BP神经网络的学习速率设 置为0.15,动量系数设置为0.9\_，最大训练次数为r# =5000;遗传算法种群初始 大小册胃=50 ,变异概率= 0.05 ,种群最多迭代7^=300次。

A-OMPN算法和BP-OMPN算法训练时间的结果如表4.4所示：

表4.4 A-OMPN和BP-OMPN算法训练时间对比表

|  |  |  |
| --- | --- | --- |
| 数据集大小（条） | A-OMPN 算法（s) | BP-OMPN 算法（s) |
| 100 | 742.260 | 38.665 |
| 200 | 2934.840 | 156.076 |
| 500 | 18513.945 | 968.742 |
| 1000 | 59564.372 | 3970.958 |
| 2000 | 335082.875 | 16484.161 |

由表格4.4画出A-OMPN算法和BP-OMPN算法训练时间的结果对比折线图如 图4.9所示：

图4.9 A-OMPN、BP-OMPN算法训练时间折线图

由图4.9可知，BP-OMPN算法训练过程的时间消耗较低，且相对于A-OMPN算 法优势明显。这主要是由于A-OMPN算法基于种群进化对训练好的多个BP网络进

行优化，而BP-OMPN算法只进行单一网络的训练，且没有遗传操作等寻优过程，因 此A-OMPN算法的训练时间较长。

综上，基于遗传神经网络的A-OMPN算法和基于BP神经网络的BP-OMPN算 法在保证了 OMPN算法较好查全率的基础上，弥补了算法查准率较低的缺陷，但是 以牺牲检测时间为代价。另一方面，A-OMPN算法的训练过程的时间消耗过大。简 化的BP-OMPN算法缩减了训练过程的时间。通过实验结果可以看出，A-OMPN算 法的效果与BP-OMPN算法的效果差距较小，因此在实际问题中，可以综合考虑精度 要求和时间要求，选择最适用于实际情况的算法。

4.4本章小结

本章首先介绍了目前较为成熟的使用遗传神经网络进行相似重复记录检测的方 式：使用训练数据集对BP神经网络进行训练，然后针对其可能陷入局部最小值的缺 点，引入遗传算法对其进行改进。然后将遗传神经网络与OMPN算法相结合提出了 A-OMPN算法，A-OMPN算法在提高查准率的同时缩减了遗传神经网络进行重复记 录检测的复杂度，但是A-OMPN算法训练网络的过程耗时严重，所以针对A-OMPN 算法，提出了简化的BP-OMPN算法，缩减了训练过程的时间消耗。最后通过对比实 验证明了 A-OMPN算法和BP-OMPN算法都可以得到较高的查全率和查准率，最后， 给出了这两个算法的训练时间，可以看出BP-OMPN算法训练较快。

第五章航天情报系统中的相似重复记录检测

目前数据清洗技术在各行业的信息管理系统中取得了广泛的应用。本章首先介绍 了航天情报信息管理系统的需求分析、概要设计以及技术实现，然后重点介绍了数据 清理模块，包括数据清理模块的设计、重复记录产生的原因、OMPN算法在系统中 的应用以及该算法对数据质量的提高。

5.1系统需求分析

5.1.1系统建设背景与目标

北京空间科技信息研宄所为了提高科技化水平，实现航天情报数据的采集、处理、 分析的信息化，于2016年开展“航天情报信息管理系统”项目的研宄。该研宄以“知 识结构化、成果产品化”为目标，立足多年的情报信息数据积累，致力于打造一款功 能丰富、实用高效的情报数据信息管理系统。

5.1.2需求分析

“航天情报信息管理系统”主要面向研宄所内部研宄人员的日常办公使用，经过 项目调研与分析后将系统的总体需求概括如表5.1所示：

表5.1需求分析总结表

|  |  |  |
| --- | --- | --- |
| 运行环境需求 | Web | 操作系统为Windows XP，浏览器为Internet Explorer 8 |
| iOS | 操作系统为iOS8.0及其以上 |
| Android | 操作系统为Android4.0及其以上 |
| 功能性需求 | 1. 数据采集模块：将现有数据采集到系统中并保持和系统中的数据   格式一致，包括两种米集模式：人工在线录入以及Excel表格批量  导入。   1. 数据清洗模块：对多源数据合并导致的重复数据进行检测清理。 2. 数据检索模块：方便研究人员更加快捷地检索和查询所需信息。 3. 服务支撑模块：包括用户权限管理、数据异常下载行为监视、综   合营销平台建设。   1. 移动应用模块：开发iOS手机端和Android手机端App，是系统在   手机端的简化体现，方便研究人员随时查看相关信息。   1. 数据应用模块：在己有数据集的基础上，对数据进行统计，并进 | |

|  |  |
| --- | --- |
|  | 行可视化展示，方便研究人员更直观地分析数据。 |
| 非功能性需求 | 1. 性能需求：并发用户数2 2000，事物平均响应时间< 3.0s。 2. 稳定性需求：双机热备方案。 3. 安全性需求：网络/系统的安全监测与检查、反爬虫设计等。 |

5.2系统设计与实现

上一节对系统的需求进行了分析总结，本节主要介绍“航天情报信息管理系统” 的设计与实现方式，主要包括系统架构、数据库设计、功能模块实现等。

5.2.1系统概要设计

根据5.1节中的系统功能性需求分析，可以将本系统按照功能模块划分成6个主 要的部分，如图5.1所示：

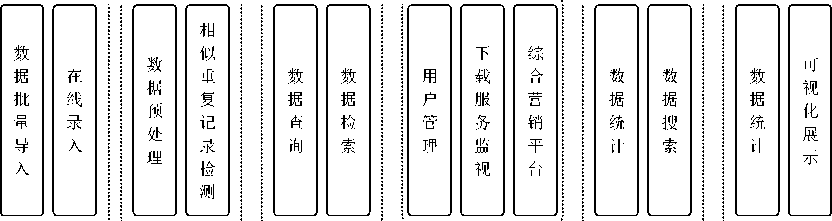
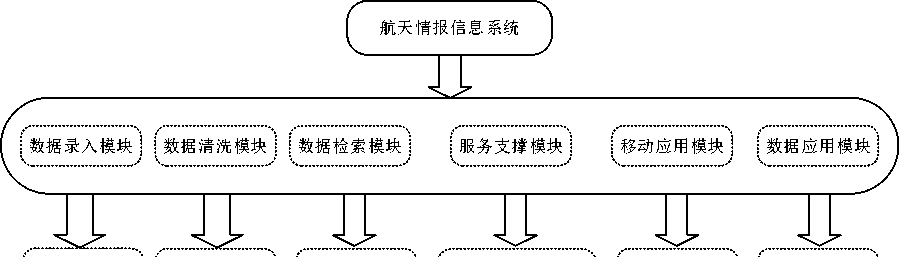


图5.1六大功能模块示意图

“航天情报信息管理系统”的总体设计从以下三个层面展开：

1. 前端和移动端的交互页面设计；
2. 服务器端逻辑功能设计；
3. 数据持久化的实现。

持久化层主要是负责数据的存储以及向服务器端提供增删改查的服务接口，这里 存储了“航天情报信息管理系统”的核心数据信息。

服务器端是处理业务逻辑的核心层，是系统的枢纽部分。数据请求由前端发给服 务器端，经用户鉴权通过之后向持久化层请求数据并进行整理发送给前端页面。

前端父互部分主要包括Web网页界面和App移动端页面，这一层是直接和用户 交互的最上一层，负责接收用户的指令以及向用户呈现系统信息等。它主要包括用户 的注册与登录、数据检索与查询、可视化展示、数据统计等功能页面。系统的总体架 构如图5.2所示：

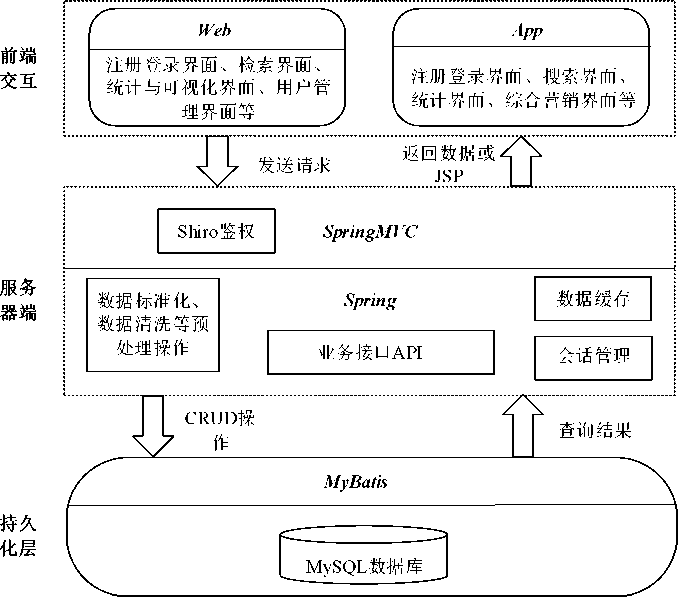


图5.2航天情报信息管理系统架构图

“航天情报信息管理系统”中的数据设计是系统设计较为核心的一个环节。数据 内容主要包括航天器、轨道信息、发射场、运载火箭、航天国家与机构、航天器故障 信息等。其中，航天器信息数据是系统的核心数据，按照所属类别又可以将其分成8 种：通信卫星、导航卫星、遥感卫星、在轨服务与空间安全卫星、空间科学卫星、技 术试验卫星、空间探测器和载人航天器。系统数据库对应的ER (实体-关系）图如图 5.3所示：

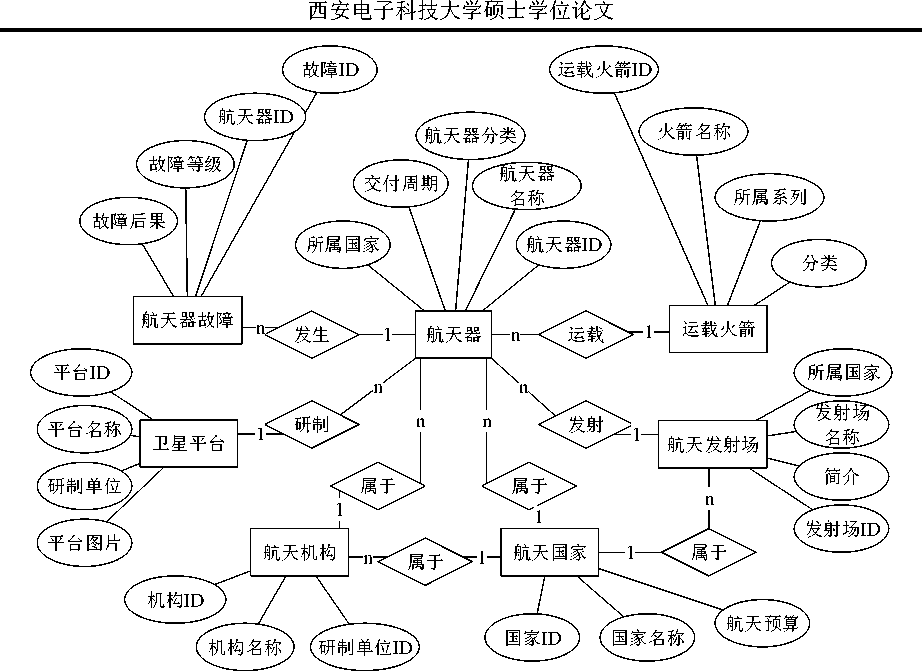


图5.3实体关系结构图

由图5.3可知，航天器实体是系统数据库中的最关键实体，它与故障、卫星平台、 航天机构、航天国家、航天发射场、运载火箭都存在直接的关系。其中，除了和航天 器故障对应关系是“一对多”之外，和其他的几个实体的关系都是“多对一”。航天 国家和与其相连的几个实体的关系均为“一对多”。

表5.2至表5.5是较为核心数据库字段设计表：

1. 航天器表

表5.2航天器字段设计表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性名称 | 属性描述 | 类型 | 可否为空 | 备注 |
| spacecraft\_id | 航天器ID | int(11) | 否 | 主键 |
| spacecraft\_name\_cn | 航天器中  文名称 | varchar(60) | 否 | 无 |
| spacecraft\_name\_en | 航天器英  文名称 | varchar(40) | 否 | 无 |
| spacecraft\_launch\_num | 航天器发  射编号 | varchar(20) | 否 | 无 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| spacecraft\_num | 航天器编  号 | varchar(20) | 否 | 无 |
| task\_property | 任务性质 | varchar(5) | 否 | 无 |
| spacecraft\_type | 航天器类  型 | varchar(50) | 否 | 无 |
| country\_id | 所属国家  ID | int(11) | 否 | 外键 |
| country | 所属国家 | varchar(20) | 否 | 无 |
| institution\_id | 所属机构  ID | int(11) | 否 | 外键 |
| institution\_name | 所属机构 | varchar(200) | 否 | 无 |
| operator\_id | 运营单位  ID | int(11) | 否 | 外键 |
| operator\_name | 所属运营  单位 | varchar(200) | 否 | 无 |
|  |  |  |  |  |
| spacecraft\_image | 外形图片 | varchar(60) | 是 | 无 |

由于篇幅限制表5.2只给出了部分关键字段的设计。从上表可以看出航天器的ID 是标识一条航天器记录的唯一关键字，而该表中的外键：country\_id、institution\_id、 operator\_id则是航天器与“所属国家”、“所属机构”以及“所属运营单位”三张表 的关联。

1. 航天国家表

表5.3航天国家字段设计表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性名称 | 属性描述 | 类型 | 可否为空 | 备注 |
| country\_id | 国家ID | int(11) | 否 | 主键 |
| country\_name\_cn | 国家中文  名称 | varchar(60) | 否 | 无 |
| country\_name\_en | 国家英文  名称 | varchar(60) | 否 | 无 |
| budget\_per\_year\_gov | 政府年度  航天预算 | float(8,3) | 是 | 无 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| budget\_per\_year\_civil | 民用年度  航天预算 | float(8,3) | 是 | 无 |
|  |  |  |  |  |
| main\_spacecraft | 主要航天  器 | text | 是 | 无 |

1. 航天器故障表

表5.4航天器故障字段设计表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性名称 | 属性描述 | 类型 | 可否为空 | 备注 |
| malfunctioned | 故障ID | int(11) | 否 | 主键 |
| malfunction\_spacecraft\_id | 故障航天器ID | int(11) | 否 | 外键 |
| malfunction\_level | 故障等级 | varchar(10) | 否 | 无 |
| malfunction\_date | 故障发生时间 | date | 否 | 无 |
| malfunction\_in\_designlife | 是否发生  在寿命期 | tinyint⑴ | 否 | 无 |
|  |  |  |  |  |
| malfunction\_consequence | 故障后果 | text | 是 | 无 |

1. 卫星平台表

表5.5卫星平台字段设计表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性名称 | 属性描述 | 类型 | 可否为空 | 备注 |
| satellite\_platform\_id | 平台ID | int(11) | 否 | 主键 |
| platform\_dev\_org\_id | 平台研制  单位ID | int(11) | 否 | 外键 |
| platform\_dev\_data | 研制时间 | date | 否 | 无 |
| platform\_descrip | 平台描述 | text | 否 | 无 |
|  |  |  |  |  |
| platform\_image | 平台图片 | varchar(60) | 是 | 无 |

5.2.2系统实现

1. 系统开发与运行环境

表5.6给出了本文所搭建的航天情报信息系统的开发环境与运行环境。

表5.6航天情报信息系统开发与运行环境

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  | App | |
|  |  | 服务器端 | Web端 | iOS | Android |
| 开发平台 | Windows 7 | | Windows 7 | macOS  Yosemite | Windows 7 |
| 开发工具 | Intellij idea  JDK 1.8.0 | | Sublime | Xcode 8 | Android Studio  JDK 1.8.0 |
| 数据库 | MySQL | |  | SQLite | SQLite |
|  |  | 操作系统： |  |  |  |
| 运行环境 | 硬  件 | Windows server  2008以上  内存大小：24G  磁盘空间：2300G | Internet  Explorer8 | iOS 8.0  及以上 | Android 4.0  及以上 |
|  |  | JDK 7.0.71 | 及以上 |
|  | 软 | Tomcat 7.0.54 |  |  |  |
|  | 件 | MySQL 5.6 |  |  |  |
|  |  | Navicat 11 |  |  |  |

1. 技术路线

本系统采用B/S加移动端C/S的综合技术方案进行实现。其中服务端采用较为成 熟的SSM (Spring、SpringMVC、MyBatis)三层技术框架，使用Maven添加依赖。 数据持久化层由MyBatis实现，它起到了对JDBC的封装作用。服务端的业务逻辑由 Spring控制，Spring框架起衔接SpringMVC和MyBatis框架作用。移动端采用经典 的MVC技术路线，Model层负责沙盒内数据的封装与维护，View层负责“空间瞭 望” App的页面展示与用户交互、包括搜索、统计、航天器信息分类浏览等， ViewController负责处理逻辑业务，如更新航天器列表、收藏航天器、统计信息提取 等，并调用Model的接口更新数据库内容。

1. 接口设计

服务端与前端以及移动端的数据传输采用JSON的数据格式，JSON数据更加简 便易读易操作，前后端交互采用Http通信协议。核心的接口设计如表5.7所示:

表5.7数据接口设计表

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 接口名称 | 请求方式 | 接口说明 |
| 1 | getHasLaunchedSpacecraft | GET | 请求己发射航天器列表 |
| 2 | getSpacecraftByCountry | GET | 按照国家分类返回  航天器列表 |
| 3 | getCountBySpacecraftType | GET | 请求某类型所有航天器数量 |
| 4 | getCountByCountry | GET | 请求某国家所有航天器数量 |
| 5 | getSearchResult | GET | 按照关键字返回检索结果 |
| 6 | getMyCollection | GET | 返回当前登录账户  收藏的航天器列表 |
| 7 | getSpacecraftDetailByID | GET | 返回某个航天器的详细信息 |
| 8 | addToCollectionByID | POST | 收藏某颗航天器 |
|  |  |  |  |
|  | login | POST | 登录 |

5.3数据清洗模块

1. “脏数据”产生原因

“航天情报信息管理系统”的数据采集方式有两种，包括人工在线填报数据以及 从现存的Excel表格数据批量导入到系统中。人工操作的出错是难以避免的，除此之 外现存Excel数据来自于不同的子部门由不同的研宄人员维护，没有统一的标准，以 上即是现存数据集中的“脏数据”产生的主要原因。再加上数据库中的航天器信息多 来自于不同的渠道，这就使得数据集中并不存在一个能唯一标识航天器的字段。

表5.7航天器重复记录举例

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 航天器  名称 | 发射场 | 发射  结果 | 发射时间 | 国家 | 研制单位 |  | 运载火箭 |
| A | Cape  Canaveral | 成功 | 2010/8/14  11:07 | 美国 | 洛克希德  -马丁 |  | 宇宙神-5 |
| A1 | 卡纳维 | 成功 | 2010.08.14 | US | 洛马 |  | Atlas-5 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 拉尔角 |  | 11:07 |  |  |  |  |
| B | 卡纳维拉尔  角发射场 | 失败 |  | United.States | Lockhead  Martin |  | 猎鹰-9 |

表5.7展示了三条航天器记录A、A1、B (因真实数据涉及商业机密故数据略有 修改）的部分信息，其中A和A1对应着同一颗航天器，B对应另外的一颗航天器。 由上表可以看出待处理的数据主要有以下特征：

1. 中英文格式不统一，如“洛克希德-马丁”和“LockheadMartin”、“宇宙神 -5” 和 “Atlas-5” 等。
2. 存在缺失数据，如B的发射时间信息缺失。
3. 中英文缩写与全拼格式不统一，如“United States”和“US”、“洛克希德- 马丁”和“洛马”等。
4. 时间格式不统一，如 A 的“2010/8/14” 和 A1 的 “2010.08.14”。

5.3.2重复记录检测算法的应用

对于“航天情报信息管理系统”所读取的原始数据，需要对其进行清洗，以去除 重复记录，这一功能在数据清洗模块进行实现。用户首先从网页端将原始数据录入系 统；然后，浏览器将数据发送到服务端进行处理；服务器端收到原始数据后，对其进 行时间格式的统一等简单操作；最后，服务器端调用数据清洗模块，实现对系统数据 的去重。本文所实现的数据清洗模块的业务流程如图5.4所示：

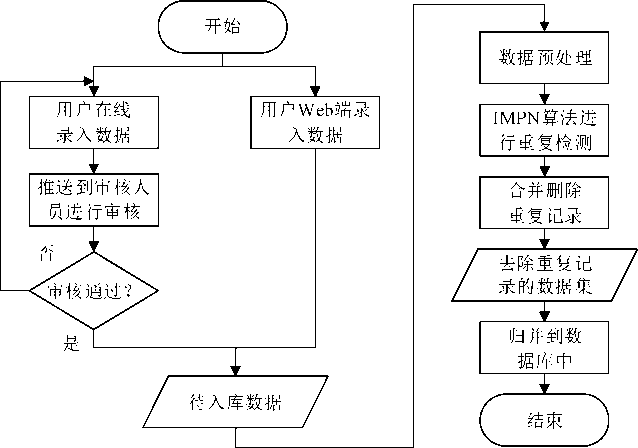


图5.4数据清洗业务流程图

在两条记录进行相似重复判断时，考虑到两条航天器同一时刻发射的概率非常小， 所以在判重过程中可以人为设定“发射时间”占有较高的权重。数据预处理过程主要 是按照固定的规则对数据进行检查，该系统所使用的预处理规则如表5.8所示：

表5.8数据预处理规则表

|  |  |
| --- | --- |
| 规则名称 | 规则描述 |
| 主键判空 | 如果记录的主键为空，则忽略该记录 |
| 曰期格式检查 | 日期统一用 yyyy-MM-dd HH:mm:ss 格式 |
| 曰期非法内容检查 | 年份超过当前年份或者当月天数超过31天，对  该记录进行标记，由人工检查处理 |
| 国家字段检查 | 国家字段的标准格式为英文全称，将中文内容  和英文缩写内容统一成标准格式 |

本文共提出了三种进行相似重复记录检测的算法，分别是OMPN算法、A-OMPN 算法和A-OMPN算法，在第三章与第四章的实验分析部分，分别给出了这三种算法 在数据生成器产生的带标签数据集上的实验结果，通过算法的查全率和查准率验证了 算法的性能。但是，数据清洗模块实际采用OMPN算法而不是A-OMPN算法或者 A-OMPN算法，其原因是：（1) “航天情报信息管理系统”中总的数据不超过8000 条，数据量比较小，OMPN算法可以满足系统对于查全率和查准率的需求。（2)数 据库中的航天器信息源自于不同的渠道，并不存在一个能唯一标识航天器的字段。所 以若采用基于遗传神经网络的A-OMPN算法或者基于BP神经网络的A-OMPN算法， 需要提取出训练数据集对网络进行迭代训练，而没有唯一标识数据记录的字段导致提 取带有标签的训练集只能由人工完成，工作量大效率较低。

5.4本章小结

本章首先介绍了“航天情报信息管理系统”的需求分析以及系统设计，然后介绍 了一些关键技术的核心实现。接着以数据清洗模块为重点进行展开，介绍了系统中的 “脏数据”的来源，数据清洗算法在系统中的应用，并设计了对比实验验证OMPN算 法在真实的航天器重复数据检测中取得的良好效果。

第六章总结与展望

6.1总结

信息时代的发展带来了剧增的数据量，由于来源的多样性、存储数据的方式和硬 件设备不同、人为错误难以避免等，导致我们获得的数据存在缺失或存在相似重复数 据，因此需要研宄出高效的算法，实现对脏数据的清洗操作。在数据清洗方面，相似 重复记录检测是一类重要的问题，本文主要研宄了相似重度记录检测问题的求解方法， 并对传统解法进行改进和创新，最后，使用本文所提出的算法实现了实际项目的数据 清洗模块，达到了系统的设计要求。本文的主要工作主要有：

1. 提出了改进的多趟近邻排序算法OMPN。在OMPN算法中，首先，提出了 基于字段区分度的关键字选取方法和基于字段区分度加权的判等方法，这种方法可以 避免传统的MPN需要依赖专家经验知识的弊端，避免了人工干扰，可以根据数据特 点提取有效地关键字。其次，提出了自适应大小的滑动窗口处理方法，在传统的MPN 算法中，需要使用固定大小的滑动窗口，带来灵活度不够、存在漏检和冗余检测的问 题，而在OMPN中，通过当前滑窗内记录的重复程度计算得到新的滑窗大小，避免 了相似度低的记录的重复检测，也增加了对相似度高的数据的检测次数，更加灵活， 效果更好。最后，传统的MPN算法对存在缺失值的数据效果较差，因此，OMPN算 法对缺失数据进行预标记，避免了数据缺失带来的不准确性。通过在不同规模的数据 集上进行实验，证明了 OMPN算法的优势。
2. 结合遗传神经网络和OMPN算法，提出增强的多趟近邻排序算法A-OMPN 和基于BP神经网络的多趟近邻排序算法BP-OMPN。遗传神经网络进行检测时，准 确度较高，但是需要对任意两个不同的记录的相似度向量进行检测，所以，检测过程 繁杂，存在冗余操作。OMPN算法提出了基于字段区分度的判等方法，但是该方法 对数据量敏感，在大规模数据上，算法的查准率下降。因此，本文将遗传神经网络与 OMPN算法相结合，对于OMPN算法的滑动窗口内的记录，使用遗传神经网络进行 判等，既避免了遗传神经网络的冗余操作，又提高了判等操作的准确度，最终得到查 准率和查全率都较高的A-OMPN算法。另外，针对遗传神经网络训练速度慢的缺点， 本文提出使用单一 BP神经网络执行判等操作，得到BP-OMPN算法。通过进行实验， 验证了 A-OMPN算法和BP-OMPN算法的查准率、查全率均较高，且BP-OMPN算 法的训练耗时明显小于A-OMPN算法。在实际问题中，可以综合考虑时间和精度要 求，选择合适的算法。
3. 在真实的航天情报信息管理系统中，运用本文提出的OMPN算法进行数据

清洗模块的搭建。该系统所提供的真实数据量为8000条数据，且真实数据没有标签， 所以使用OMPN算法进行构建。

6.2展望

本文主要研宄了数据清洗领域的相似重复记录检测问题，针对研宄过程中发现的 问题与不足之处，今后仍然需要在以下几个方面进行深入研宄：

1. 算法对于中文数据库的处理能力较弱，因为在生成排序关键字以及大小排 序时对字符的处理是基于ASCII值的，中文的相似重复记录检测一般先进行分词处理， 所以在中文重复记录检测方面有待于更广泛和深入地研宄。
2. 实验部分使用的数据量大小还远未到海量数据的标准，当数据量大小超出 内存所能容纳的上限时，提取数据集和排序过程均需要作出调整。可能的解决方案包 括将数据集划分为多个更小的单位，采用外部排序等。数据量增大时，算法的效率问 题也是亟待考察和解决的。