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究与创新。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。

本文首先研究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，通过研究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。

3. 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

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, it's 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.

2. 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.

3. 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 研究的背景和意义

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究重点。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[2][3]。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，所以研究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究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

据集进行整合被认为是一个困难而且极为重要的问题，最早的研究主要是从数据连接[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、等领域中的研究重心之一。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究的开端[9]。

数据清洗的研究重点包括：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，主要的工作有两个方面——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，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。复旦大学周傲英等比较早开始数据清理的研究工作[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究机构开发的数据清洗软件[22][23]。

在相似重复记录检测领域，国内外研究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，所以仍然有研究的价值和改进的空间。

1.3 论文研究的主要内容

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，一直是数据清洗研究的重点问题。

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。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究过程中遇到的问题，并对未来的研究方向进行了展望。

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 R_1 、 R_2 ，它们的内容相同或者相似，且都对应着同一个现实实体 E ，则记录对 $< R_1, R_2 >$ 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.Sam W”，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 S_1 和 S_2 的编辑距离是指：S_1 变成 S_2 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 S_1 和 S_2 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。

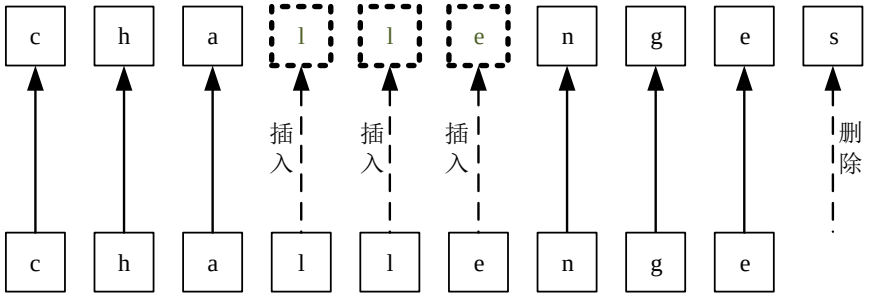

图 2.1 编辑距离示意图

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 S_1 和 S_2，两者的 Jaro 距离如下公式(2-1)所示[27]：

$$D_{Jaro}=\frac{1}{3}(\frac{m}{|S_1|}+\frac{m}{|S_2|}+\frac{m-t}{m}) \tag{2-1}$$

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：

$$D_{J-W}=D_{jaro}+L*P(1-D_{jaro}) \tag{2-2}$$

在公式(2-2)中，D_{jaro} 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，$P\le 0.25$。

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 $w_i(i=0,1,\cdots n)$；

2.对每个单词 w_i 分配权重，$weight_i=\log(tf_{w_i}+1)*\log(idf_{w_i})$，其中单词出现的次数（词频）用 tf_{w_i} 表示，idf_{w_i} 表示记录总数除以包含 w_i 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 b；

4.计算向量的余弦相似度：$\cos\theta=\dfrac{\sum_{i=1}^{n}(A_i\times B_i)}{\sqrt{\sum_{i=1}^{n}(A_i)^2}\times\sqrt{\sum_{i=1}^{n}(B_i)^2}}$；

5. $\cos\theta$ 值越接近 1，则说明相似度越高，将 $\cos\theta$ 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 $O(N^2)$，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_i 的排序关键字为 key_i；然后，按照记录的 key_i 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 R_i 生成排序关键字 key_i；

3.按照 key_i 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：

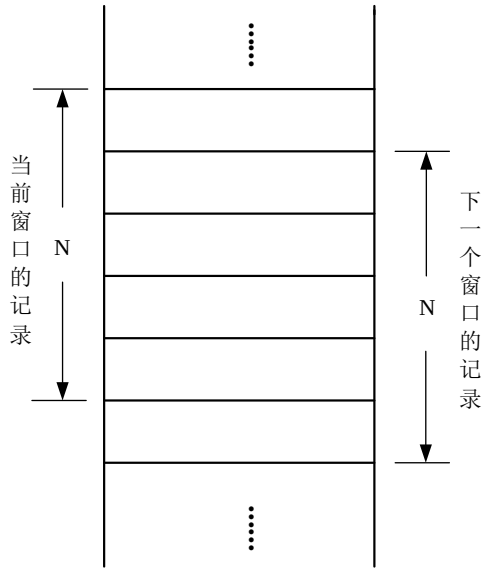

图 2.2 SNM 算法滑动窗口过程示意图

若数据集大小为 N，使用 SNM 算法生成排序关键字过程的时间复杂度为 $O(N)$，本文使用快速排序，其时间复杂度为 $O(N\log N)$，对滑动窗口内记录的判等操作的时间复杂度为 $O(wN)$，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[30]是：设 R 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 R_1 称为 R 的传递闭包：

（1）$R \subseteq R_1$；

（2）R_1 满足性质 P；

（3）如果存在集合 A 上的关系 R'，R' 满足性质 P 并且 $R' \supseteq R$，则 $R_1 \subseteq R'$。

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 k:= 1 to n

  FOR i := 1 to n

    FOR j := 1 to n

      $W[i, j] \leftarrow W[i, j] \vee (W[i,k] \wedge W[k, j])$

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 R_1 和 R_2 是相似重复记录，记录 R_2 和 R_3 是相似重复记录，则 R_1 和 R_3 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 m 个排序关键字的生成方案，独立重复地执行步骤 2~4 m 次；

2.对数据集生成排序关键字；

3.按照 $key_i (i = 1, \cdots, m)$ 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 $w_i (i = 1, \cdots, m)$，每次比较时将新进入窗口的记录 R_w 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 $D_i (i = 1, \cdots, m)$；

5.对 m 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：

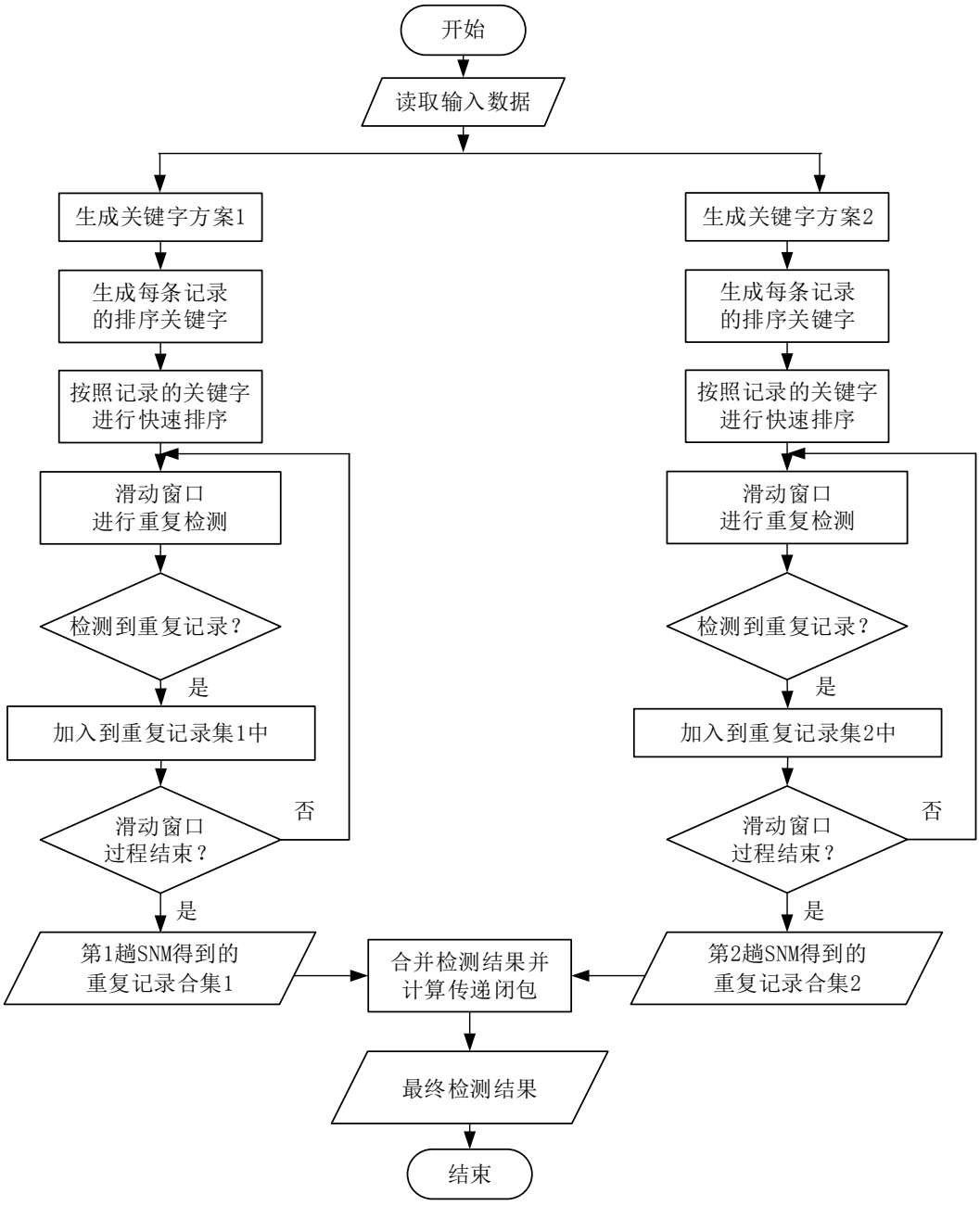

图 2.3 MPN 算法流程图

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 R_1 、 R_2 的 N-Gram 相似性计算方法如公式(2-3)所示[20]：

$$sim_q(R_1,R_2)=\frac{\left|G_q(R_1)\cap G_q(R_2)\right|}{\left|G_q(R_1)\cup G_q(R_2)\right|} \tag{2-3}$$

公式(2-3)中，$sim_q(R_1,R_2)$ 表示记录 R_1 、R_2 的 N-Gram 相似性，$G_q(R)$ 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]：

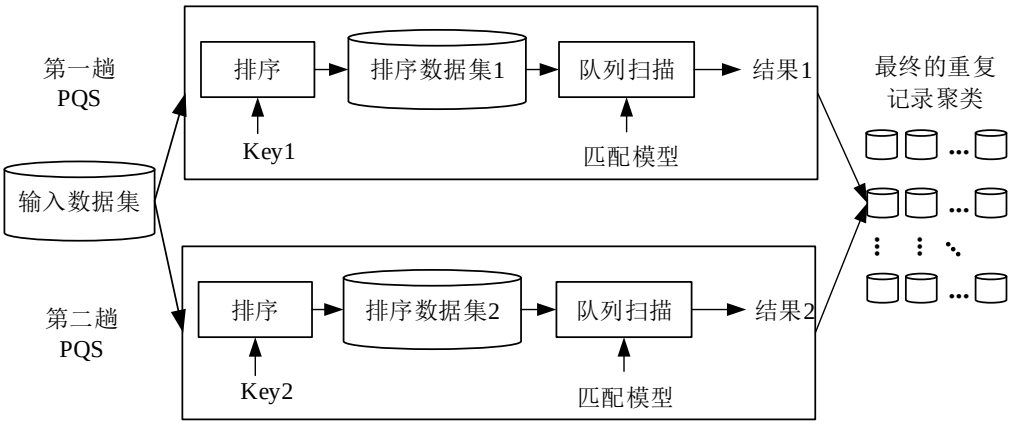

图 2.4 多趟优先队列扫描算法过程示意图

2.4 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]：

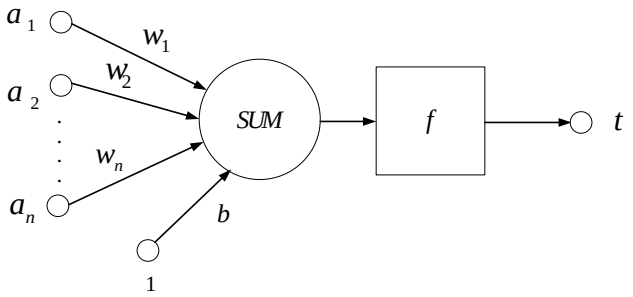

图 2.5 神经元模型

其中，a_i 是输入向量的第 i 个分量，w_i 是第 i 个输入分量连接到该神经元的权值，b 表示偏置，SUM 是加权求和操作，f 表示激活函数。则对于输入向量 $input=(a_1,a_2,...,a_n)$ 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：$output=f(\sum_{i=1}^{n}w_ia_i+b)$。许多的类似于这样的神经元则组成了人工神经网络。

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，$\{\theta_0,\theta_1,...,\theta_d\}$ 是要学习的参数，x_i^j 是第 j 个输入特征向量的第 i 个分量，$x_0^j=1$，θ_0 表示偏置，共有 d 维特征。

$$h_\theta(x^j)=\theta_0+\theta_1x_1^j+\theta_2x_2^j+...+\theta_dx_d^j=\sum_{i=0}^{d}\theta_ix_i^j \tag{2-4}$$

采用均方误差损失函数，如公式(2-5)所示[36]，共有 m 个训练样本，y^j 表示第 j 个训练样本的真实类标向量，$h_\theta(x^j)$ 表示第 j 个训练样本的预测类标向量，当损失函数 $J(\theta)$ 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 $J(\theta)$[36]。

$$J(\theta)=\frac{1}{2}\sum_{j=1}^{m}\left(y^j-h_\theta(x^j)\right)^2 \tag{2-5}$$

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 $J(\theta)$ 对参数 θ_i 的负梯度方向更新参数 θ_i，如公式(2-6)所示。η 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 η 逐渐减小，即参数在学习过程中的变化越来越小。

$$\frac{\partial J(\theta)}{\partial \theta_i} = -\sum_{j=1}^{m}\left(y^j - h_\theta(x^j)\right)\bullet\frac{\partial h_\theta(x^j)}{\partial \theta_i} = -\sum_{j=1}^{m}\left(y^j - h_\theta(x^j)\right)\bullet x_i^j \tag{2-6}$$

$$\theta_i \leftarrow \theta_i + \eta\left(-\frac{\partial J(\theta)}{\partial \theta_i}\right) = \theta_i + \eta\sum_{j=1}^{m}\left(y^j - h_\theta(x^j)\right)\bullet x_i^j \tag{2-7}$$

由公式(2-7)可以看出，对每一个参数 $\theta_i\left(i=1,2,..,d\right)$，都需要使用全部样本来学习该参数的变化量，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

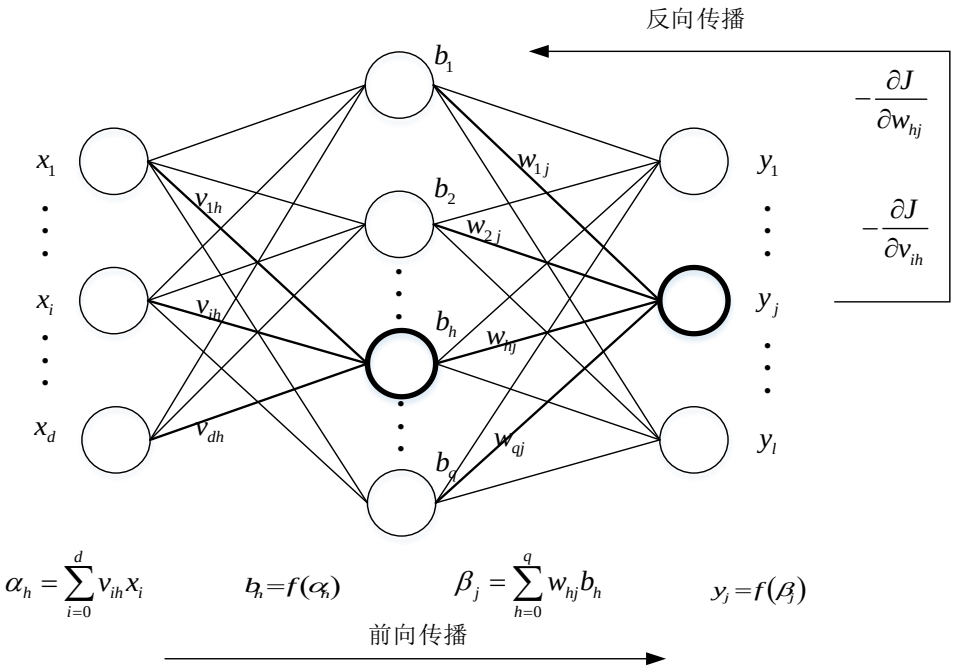

图 2.6 典型三层 BP 神经网络

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 q 个隐层神经元构成第二层，第三层包含 l 个输出神经元，表示输出层[38,39,40,41]。v_{ih} 表示第 i 个输入神经元与第 h 个隐层神经元的连接权值，v_{0h} 表示第 h 个隐层神经元

的偏置，w_{hj} 表示第 h 个隐层神经元与第 j 个输出神经元的连接权值，w_{0j} 表示第 j 个输出神经元的偏置，x_i 表示第 i 个输入神经元的输入特征值，b_h 表示第 h 个隐层神经元的输出，y_j 表示第 j 个输出神经元的输出，α_h 表示第 h 个隐层神经元的输入，β_j 表示第 j 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 $f(\bullet)$ 进行处理，得到下一层对应神经元的输出值，$f(\bullet)$ 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]。

$$f(z)=\frac{1}{1+e^{-z}} \tag{2-8}$$

$$f(z)=\frac{1-e^{-2z}}{1+e^{-2z}} \tag{2-9}$$

$$f(z)=\max(0,z) \tag{2-10}$$

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 $f(\bullet)$ 使用 sigmoid 函数。

$$b_h=f\left(\sum_{i=0}^{d}v_{ih}x_i\right) \tag{2-11}$$

$$y_l=f\left(\sum_{h=0}^{q}w_{hl}b_h\right) \tag{2-12}$$

对于第 k 个训练样本 $\left(x^k,y^k\right)$，$x^k=(1,x_1^k,x_2^k,...,x_d^k)$，$y^k=(y_1^k,y_2^k,...,y_l^k)$，设神经网络对于该样本的输出为 $\overline{y^k}=(\overline{y_1^k},\overline{y_2^k},...,\overline{y_l^k})$，使用均方误差损失函数，则对样本 $\left(x^k,y^k\right)$ 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 L_2 正则项，W 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。

$$J_k=\frac{1}{2}\sum_{j=1}^{l}\left(\overline{y_j^k}-y_j^k\right)^2 \tag{2-13}$$

$$J=\frac{1}{m}\sum_{k=1}^{m}J_k+\frac{1}{2}\|W\|^2 \tag{2-14}$$

$$f'(z)=f(z)\left(1-f(z)\right) \tag{2-15}$$

根据链式求导法则[36]，损失函数 J_k 对参数 w_{hj} 和 v_{ih} 的导数分别如公式(2-16)和公式(2-17)所示。

$$\frac{\partial J_k}{\partial w_{hj}}=\frac{\partial J_k}{\partial \overline{y_j^k}}\frac{\partial \overline{y_j^k}}{\partial \beta_j}\frac{\partial \beta_j}{\partial w_{hj}}=\left(\overline{y_j^k}-y_j^k\right)\overline{y_j^k}\left(1-\overline{y_j^k}\right)b_h \tag{2-16}$$

$$\begin{aligned}\frac{\partial J_k}{\partial v_{ih}}&=\left(\sum_{j=1}^{l}\frac{\partial J_k}{\partial \overline{y_j^k}}\frac{\partial \overline{y_j^k}}{\partial \beta_j}\frac{\partial \beta_j}{\partial b_h}\right)\frac{\partial b_h}{\partial \alpha_h}\frac{\partial \alpha_h}{\partial v_{ih}}\\&=\left(\sum_{j=1}^{l}\left(\overline{y_j^k}-y_j^k\right)\bullet f(\beta_j)\left(1-f(\beta_j)\right)\bullet w_{hj}\right)\bullet f(\alpha_h)\left(1-f(\alpha_h)\right)\bullet x_i\\&=b_h\left(1-b_h\right)x_i\sum_{j=1}^{l}\left(\overline{y_j^k}-y_j^k\right)\overline{y_j^k}\left(1-\overline{y_j^k}\right)w_{hj}\end{aligned} \tag{2-17}$$

在样本 $\left(x^k,y^k\right)$ 上，参数的更新如公式(2-18)所示。

$$\begin{aligned}w_{hj}&\leftarrow w_{hj}-\eta\frac{\partial J_k}{\partial w_{hj}}\\v_{ih}&\leftarrow v_{ih}-\eta\frac{\partial J_k}{\partial v_{ih}}\end{aligned} \tag{2-18}$$

BP 误差反向传播算法的基本过程为：

输入：训练集 $D=\left\{\left(x_k,y_k\right)\right\}_{k=1}^{m}$，学习率 η.

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

For all $\left(x_k,y_k\right)\in D$：

根据当前参数执行前馈传播，计算当前样本的输出值 $\overline{y^k}$；

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
R_1	Jack	Stolfo	123 First Street	12345678
R_2	Jack	Stolfo	123 First Street	12345673
R_3	Jack	Stolpho	123 First Street	12345678
R_4	Jacon	Stiles	123 Forest Street	12345432

表 3.1 列举了四条记录 R_1、R_2、R_3 和 R_4，它们由四个字段组成，分别是 First Name、Last Name、Address 和 ID，其中，R_1、R_2、R_3 互为重复记录，R_4 与其他记录不同。记录 R_2 和 R_1 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。R_3 和 R_1 的 Last Name 字段不一致，这也是由于人为错误导致的。而 R_4 和 R_1 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 $R_1 = R_2 = R_3 \neq R_4$ 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）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

方式 2、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

（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

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_1	Jack	Stolfo	123 First Street	12345678	STOLFO123FRT123
R_2	Jack	Stolfo	123 First Street	12345673	STOLFO123FRT123
R_3	Jack	Stolpho	123 First Street	12345678	STOLPHO123FRT123
R_4	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_1	Jack	Stolfo	123 First Street	12345678	STLJAC123FRT213
R_2	Jack	Stolfo	123 First Street	12345673	STLJAC123FRT123
R_3	Jack	Stolpho	123 First Street	12345678	STLJAC123FRT123
R_4	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_1	Jack	Stolfo	123 First Street	12345678	STLJCK123FRT213
R_2	Jack	Stolfo	123 First Street	12345673	STLJCK123FRT123
R_3	Jack	Stolpho	123 First Street	12345678	STLJCK123FRT123
R_4	Jacon	Stiles	123 Forest Street	12345432	STLJCN123FRT123

由表 3.2 可以看出，记录 R_1 和 R_2 对应的关键字相同，均为“STOLFO123FRT123”，记录 R_1 和 R_4 对应的关键字不同，经过排序后 R_1 和 R_2 会聚集在邻近位置，而 R_1 和 R_4 则不会处于邻近的位置。对于记录 R_1 和 R_3，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 R_1 和 R_3 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_1 和 R_2 具有相同排序关键字是合理的；R_1 和 R_3 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_1 和 R_3 的排序关键字的相同也是合理的；然而，R_4 和 R_1 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 R_4 和 R_1 具有相同的排序关键字，经排序后，R_4 和将会与 R_1、R_2、R_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 R_4 与 R_1、R_2、R_3 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_1、R_2、R_3 的排序关键字相同，它们对应的现实实体也是一致的，R_4 和 R_1 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 N 条记录，每条记录都 M 个字段组成，即 $fields=\{field_1, field_2, \ldots, field_m\}$，对于第 i 个字段 $field_i$，它的区分度计算公式如公式(3-1)所示：

$$diff_{field_i}=\frac{x_{field_i}}{N} \tag{3-1}$$

其中，x_{field_i} 代表数据集中在 $field_i$ 字段一共有 x_{field_i} 种取值，即如果将数据集按照字段 $field_i$ 的不同取值进行聚类，一共有 x_{field_i} 簇。$diff_{field_i}$ 的取值介于 0 到 1 之间，$diff_{field_i}$ 值越高，说明对应的字段 $field_i$ 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 m 个组成部分，以及每个部分的生成方案 $key_i = f_i(string), i = 1,...,m$；

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 $f_i(string)$ 生成有 m 个组成部分的排序关键字 $\{key_1, key_2, ...key_m\}$；

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_1、R_2（均包含 num 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_1、R_2，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_1、R_2 的调整后的编辑距离向量 $edit$；

（3）按照公式(3-2)计算赋给第 i 个字段的权值 $weight_i$；

$$weight_i = \frac{i^2}{\sum_{j=1}^{num} j} \tag{3-2}$$

（4）按照公式(3-3)计算记录 R_1 、 R_2 的相似度 sim；

$$sim = \sum_{i=1}^{num}(edit_i \times weight_i) \tag{3-3}$$

（5）将 sim 和阈值 T 进行比较，若 $sim \geq T$，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 R_1 和 R_2，计算记录 R_1 和 R_2 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 $weight_i$；

4.计算 R_1 和 R_2 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。

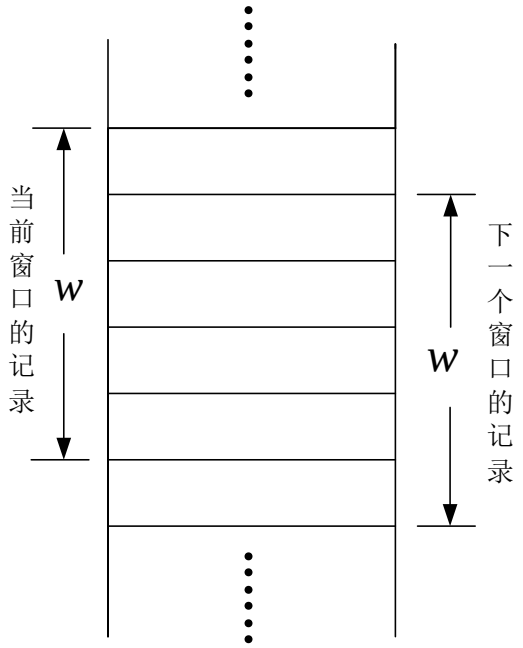

图 3.1 滑动窗口扫描过程

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 $O(wN)$，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 $w_2=5$，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。

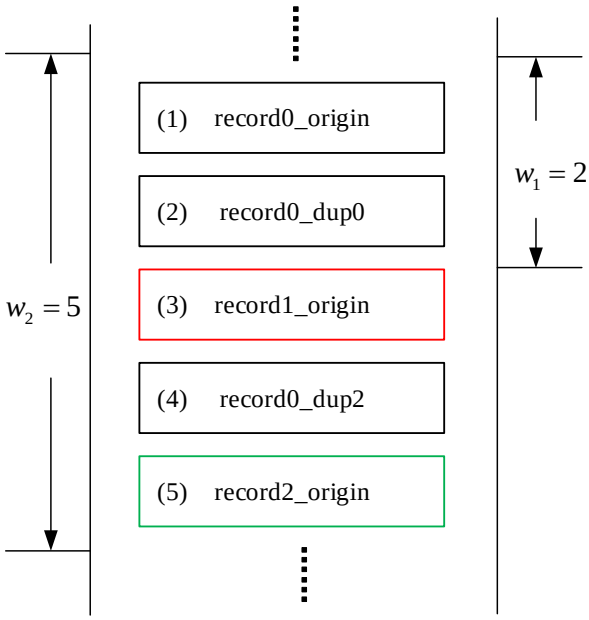

图 3.2 不同滑动窗口大小时的归并过程

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 $O(wN)$，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 $O(N^2)$。

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：

$$w_{next} = w_{\min} + (w_{\max} - w_{\min}) \times \frac{\sum_{i=index-w+1}^{index-1} |i - index| \times B_i}{\sum_{i=index-w+1}^{index-1} |i - index|} \tag{3-4}$$

公式(3-2)中，常数 $W_{\max}$ 表示滑动窗口的大小可以取的最大值，常数 $W_{\min}$ 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，B_i 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 B_i =1，否则 B_i =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 $w_{\max}$ ，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 $w_{\min}$ ；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）。

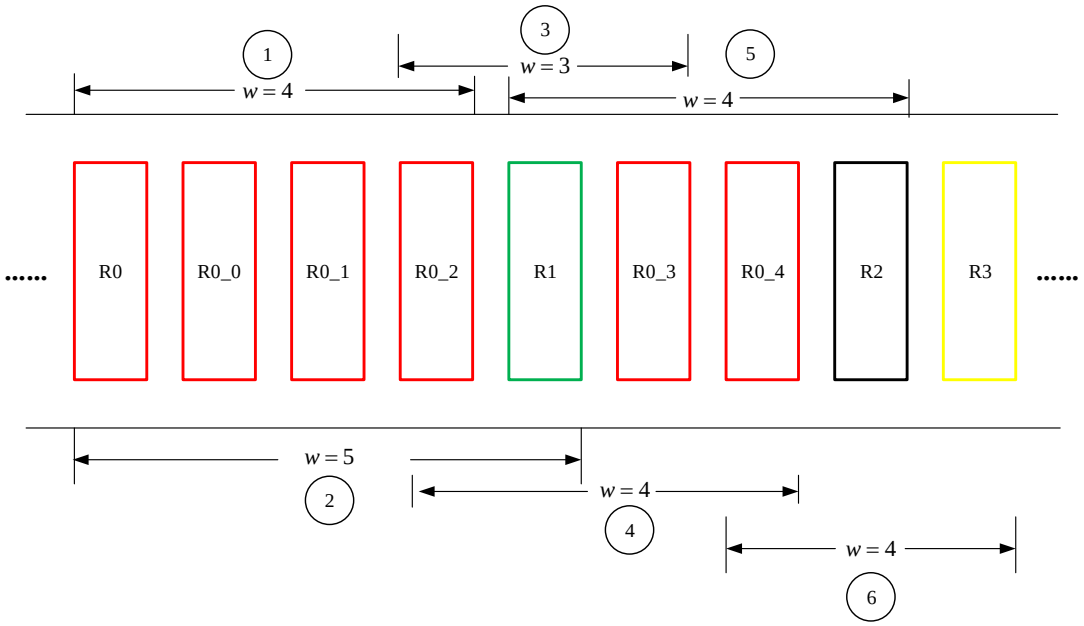

图 3.3 自适应大小的滑动窗口

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

3.3.1 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_1	Jack	tolfo	123 First Street	12345678
R_2	Jack		123 First Street	12345673
R_3	Jack	Stolpho	123 First Street	12345678
R_4	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 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究领域，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 进行标记；

（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。

3.4 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 N 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 N 组排序关键字 $\{key_1, key_2, ...key_n\}$；

4.独立地执行步骤 5~8 N 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 key_i；

6.对数据集按照关键字 key_i 排序，如果某条记录的 key 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 $set\_with\_incomplete\_key_i$ 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 dup_set_i；

8.将 dup_set_i 与 $set\_with\_incomplete\_key_i$ 进行重复归并，然后计算此集合的传递闭包 $transitive\_closure\_set_i$；

9.将 N 次 SNM 重复检测得到的 $transitive\_closure\_set_i$ 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 $total\_dup\_set$。

含有两趟 SNM 过程的 OMPN 算法流程如图 3.4 所示：

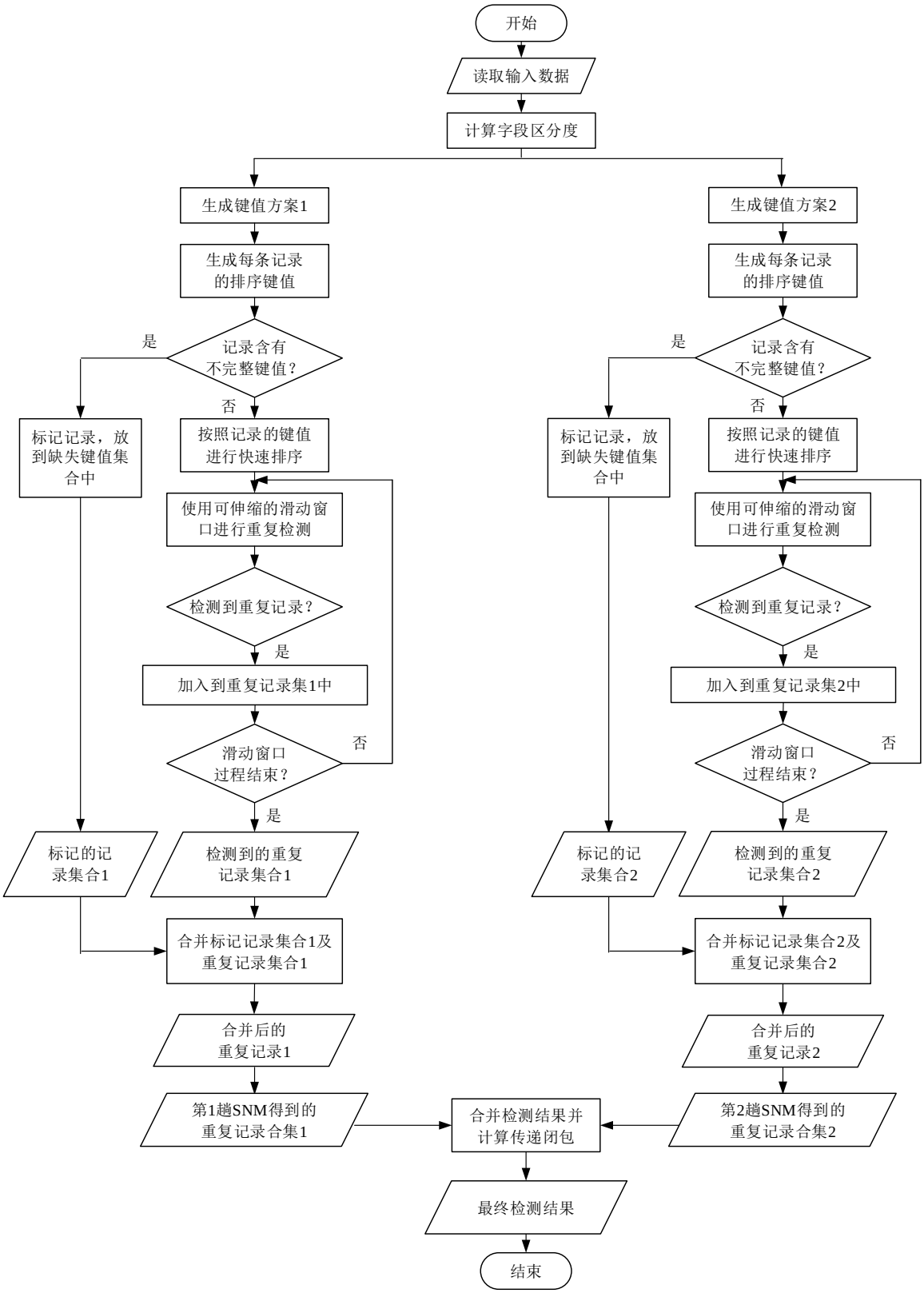

图 3.4 含有 2 趟 SNM 过程的 OMPN 算法流程图

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$O(N)$；排序过程采用快速排序，算法时间复杂度为$O(N\log N)$；滑动窗口的归并检测过程需要进行ωN次比较，所以该阶段的时间复杂度为$O(\omega N)$；其中N代表待检测数据集中的记录总数，ω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d，则该阶段的时间复杂度为$O(d^3)$。所以对于 MPN 算法来说，总的时间复杂度为：

$$T_{mpn}=O(N)+O(N\log N)+O(\omega N)+O(d^3) \tag{3-5}$$

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k，则区分度统计阶段的时间复杂度为$O(kN)$；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$O(N)$、$O(N\log N)$、$O(\omega N)$；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$O(d^3)$，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$m>\omega$）：

$$T_{IMPN}=O(kN)+O(N\log N)+O(\omega N)+O(mN)+O(d^3) \tag{3-6}$$

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a代表常数，两个公式都可以简化成下面的式子：

$$T=O(aN)+O(N\log N)+O(d^3) \tag{3-7}$$

当重复记录较多时，即d的值较大，此时$O(d^3)>O(aN)+O(N\log N)$两个算法的时间复杂度均为$O(d^3)$，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。

3.5 SNM、MPN、OMPN 综合对比实验

3.5.1 实验数据介绍

为了方便研究使用，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”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_1	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（*recall*）、查准率（*precision*）以及 *F-measure*[13][14]。

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）True Positive（*TP*）：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（*TN*）：算法判定为非重复记录，实际也不是重复记录；

（4）False Negative（*FN*）：算法判定为非重复记录，但实际上却是重复记录。

查全率（*recall*）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：

$$recall = \frac{TP}{TP + FN} \tag{3-8}$$

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：

$$precision = \frac{TP}{TP + FP} \tag{3-9}$$

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)：

$$F_{\alpha}=\frac{(\alpha^2+1)precision\times recall}{\alpha^2 precision+recall} \tag{3-10}$$

其中参数α取值为 1 时，*F-measure* 即为最常见的 *F1-measure*：

$$F_1=\frac{2precision\times recall}{reecision+recall} \tag{3-11}$$

假设有 8 条记录 R_1、R_2、R_3、R_4、R_5、R_6、R_7、R_8，其中{ R_1， R_2， R_3}和{ R_4， R_5， R_6}互为相似重复记录，若通过算法检测出的重复结果为{ R_1， R_2， R_3， R_7}和{ R_4， R_5， R_6， R_8}互为相似重复记录，则 *TP*=4，*FP*=2，*TN*=2，*FN*=0，所以算法的查全率为 4/(4+0)=100.00%，查准率为 4/(4+2)=66.67%，*F1-measure*=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 $\omega_{\max}$=20，最小为 $\omega_{\min}$=3，进行单趟 SNM 的次数 t=3；

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

SNM 算法的实验参数：固定滑动窗口大小为 ω=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	p_1	p_2	p_3	p_4	p_5	p_6	p_7
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 $p_1 \sim p_7$ 代表的含义分别为：p_1-原始记录数、p_2-重复记录数、p_3-单个记录最多重复数、p_4-单个字段最多修改数、p_5-单个记录最多修改字段数、p_6-重复记录的概率分布、p_7 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 *recall* 、 *precision* 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

表 3.12 SNM、MPN、OMPN 算法运行时间对比表

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

根据表 3.10-表 3.12 的实验结果，分别作出查准率、查全率和运行时间的折线图，如图 3.5 至图 3.7 所示。

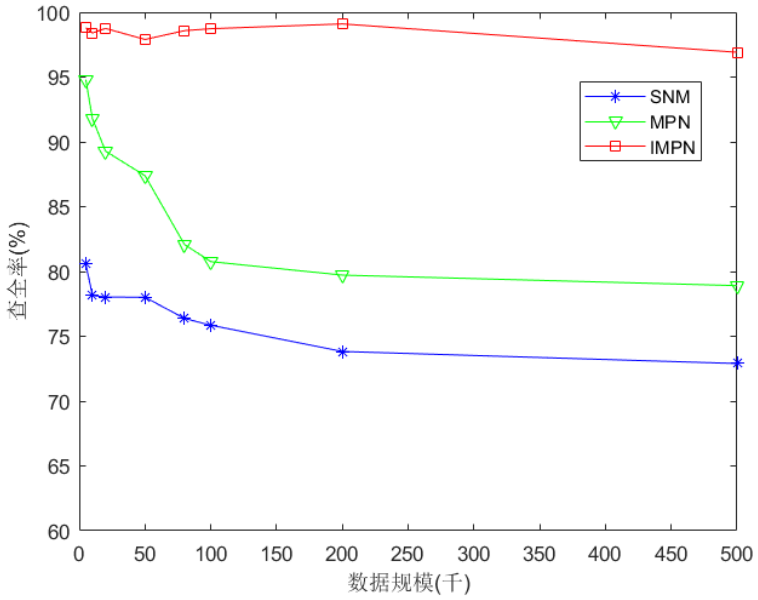

图 3.5 SNM、MPN、OMPN 算法查全率折线图

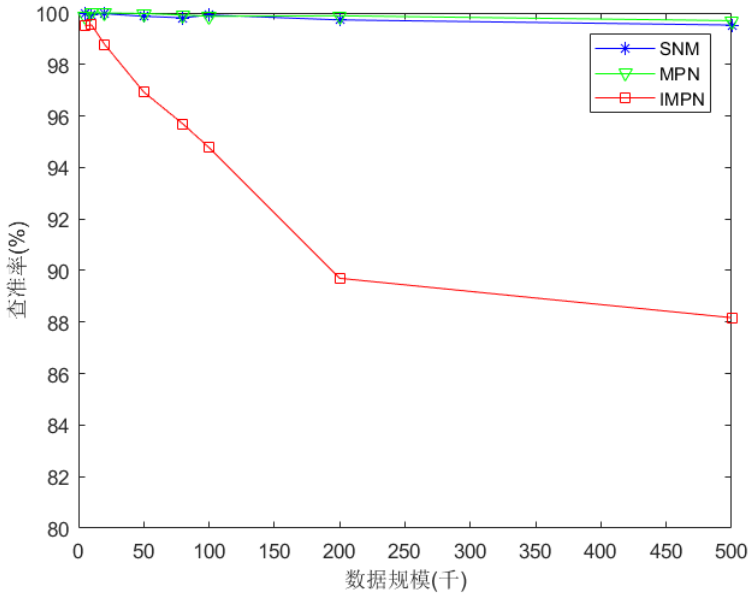

图 3.6 SNM、MPN、OMPN 算法查准率折线图

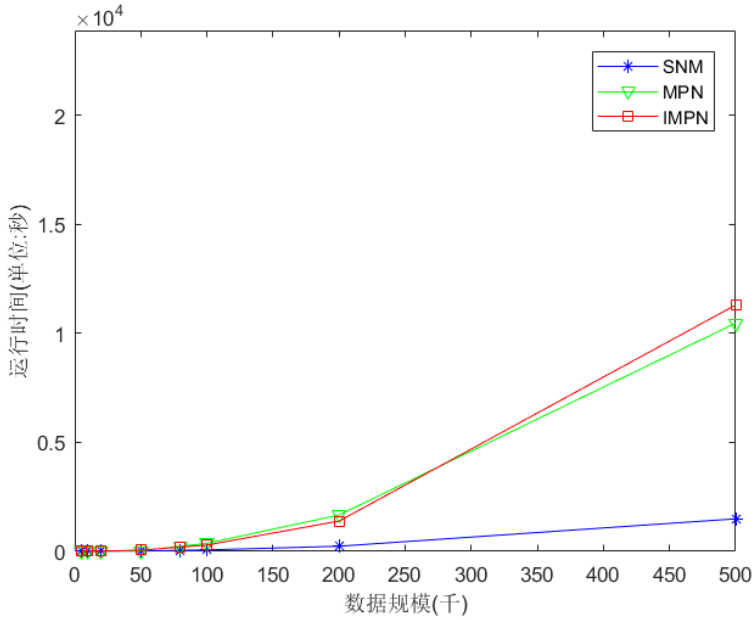

图 3.7 SNM、MPN、OMPN 算法运行时间折线图

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

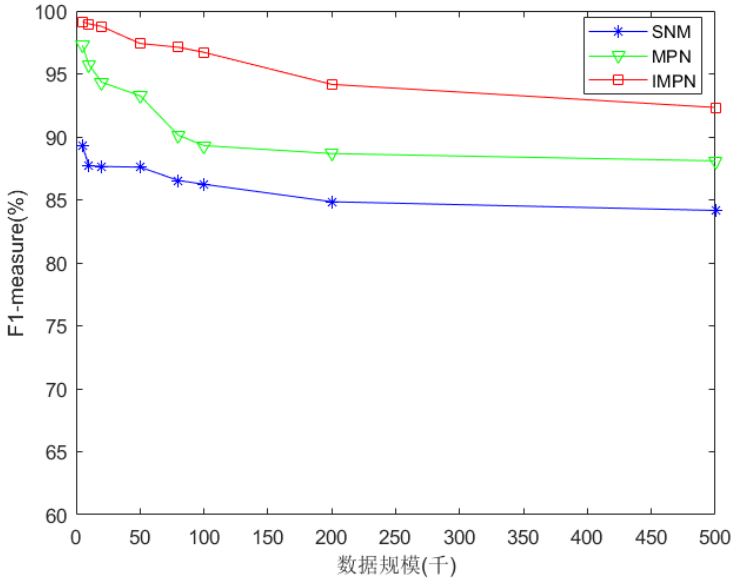

图 3.8 SNM、MPN、OMPN 算法 F1-measure 折线图

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）。

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

4.1.1 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。

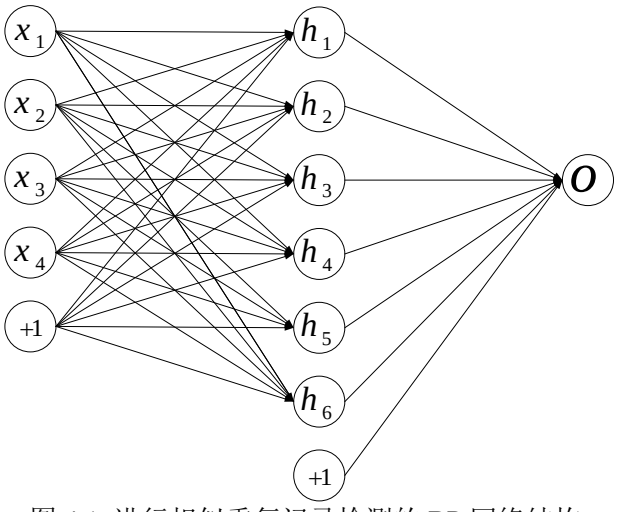

图 4.1 进行相似重复记录检测的 BP 网络结构

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 m。任取两条记录，计算这两个记录在第 i 个字段下的相似度值为 sim_i，$i=1,2,...m$，则这两条记录的相似度向量为 $(sim_1, sim_2, ..., sim_m)$，BP 网络的输入层各结点的取值也随之确定，即第 i 个结点的输入为 sim_i。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_id”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。

$$\begin{aligned} M_1 &= \sqrt{0.43NK + 0.12K^2 + 2.54N + 0.77K + 0.35} + 0.51 \\ M_2 &= \begin{cases} N + 0.618(N - K), N \geq K \\ N - 0.618(K - N), N < K \end{cases} \\ M &= \max(M_1, M_2) \end{aligned} \tag{4-1}$$

其中 M 代表隐含层结点的个数，N 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 N =15，通过公式(4-1)计算得到 M_1 =7，M_2 =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：

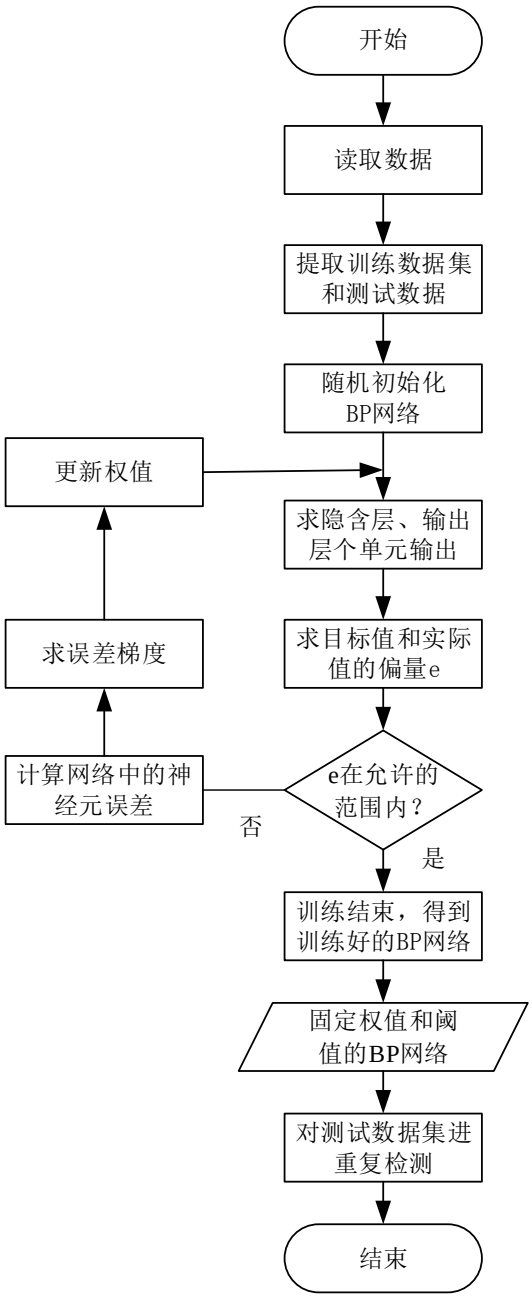

图 4.2 BP 网络检测重复记录的流程图

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 P，则共有 P 组神经网络权值和阈值，即需要训练 P 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。

2.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 i 个个体的适应度值。

$$f_i = \frac{1}{1+E_i} \tag{4-2}$$

公式(4-2)中，E_i 代表使用第 i 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_i 的计算公式如公式(4-3)所示。

$$E_i = \sum_{j=1}^{N}(g_j - a_j)^2 \tag{4-3}$$

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

3.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：

$$P_i = \frac{f_i}{\sum_{i=1}^{M} f_i} \tag{4-4}$$

其中，M 代表种群中个体的数量，f_i 代表第 i 个染色体的适应度函数的值。

4.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 P 组 BP 网络；然后，用这 P 组 BP 网络的权值产生初始状态下的种群，种群中含有 P 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$O(N^2)$，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。

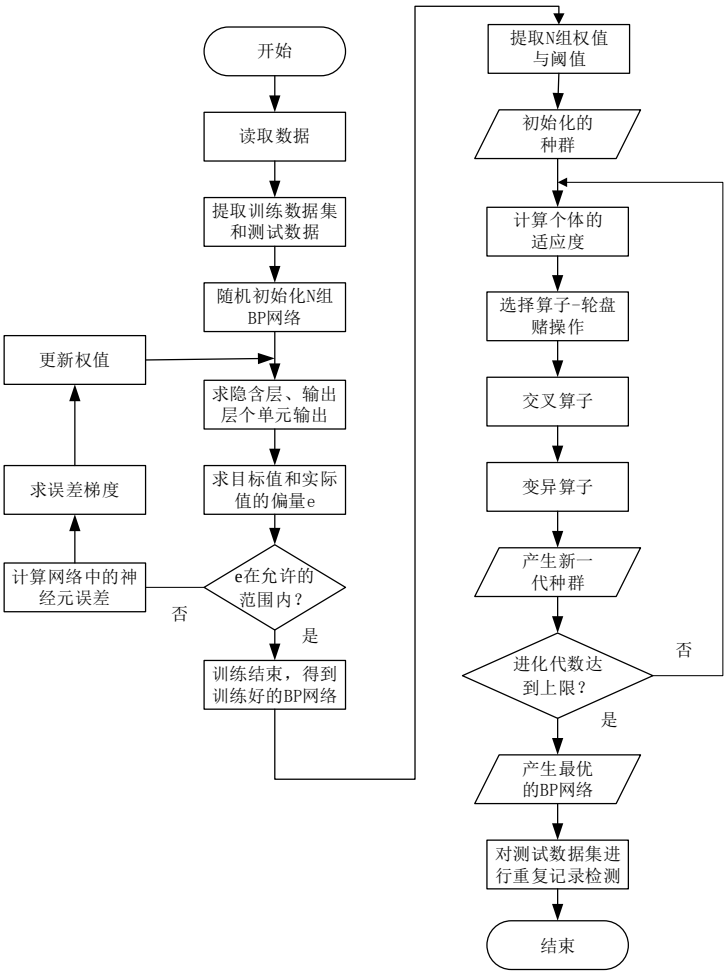

图 4.3 遗传神经网络检测相似重复记录流程图

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$O(N^2)$。

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m_{size}个 BP 神经网络，然后用训练得到的m_{size}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m_{size}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t_{SNM}；

2.随机产生m_{size}个初始 BP 神经网络；

3.分别对m_{size}个 BP 网络进行训练，用训练完成的m_{size}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BP_{best}；

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D_1中；

7.对数据全集中的、非D_1中的其余记录进行排序；

8.采用BP_{best}进行可伸缩大小的滑动窗口重复检测，得到重复集合D_2；

9.在集合D_2中检测D_1中的记录的重复记录，得到最终的重复记录集合D；

10.独立地执行步骤 5~9t_{SNM}次，得到t_{SNM}个重复记录集合，并对t_{SNM}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D_F；

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。

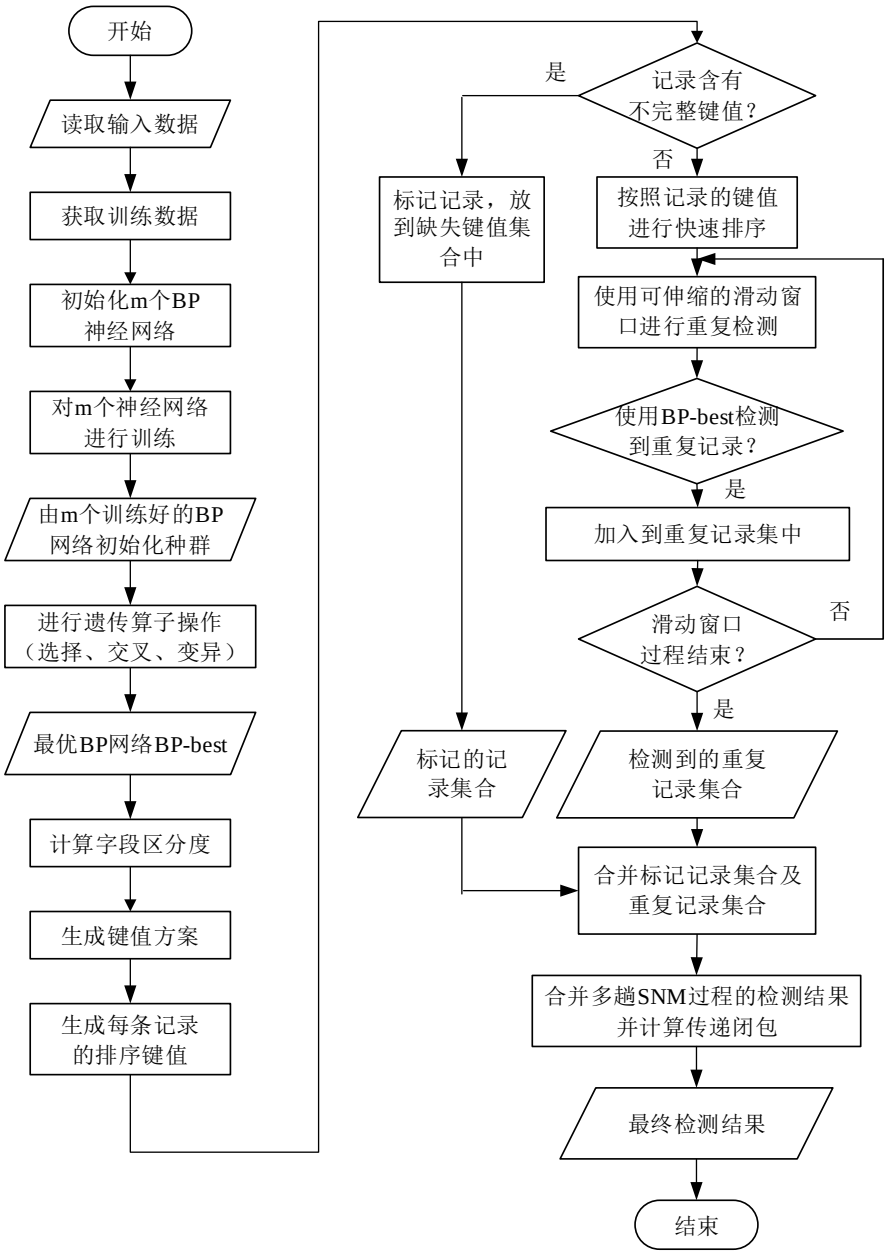

图 4.4 A-OMPN 算法流程图

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n_I，隐含层节点个数为n_H，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$O(n_I n_H)$，设 BP 网络终止训练过程的最大次数为T_{BP}，遗传算法的种群

数量为m_{size}，则需要训练m_{size}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$O(m_{size}n_In_H)$，种群演化过程的时间复杂度为$O({m_{size}}^2)$，所以 A-OMPN 算法训练过程的总时间复杂度如公式(4-8)：

$$T_{A-IMPN\_train}=\begin{cases}O(m_{size}n_In_H), n_In_H \geq m_{size}\\ O({m_{size}}^2), \quad n_In_H < m_{size}\end{cases} \tag{4-5}$$

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t_{SNM}；

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D_1中；

5.对数据全集中的、非D_1中的其余记录进行排序；

6.采用BP_{best}进行可伸缩大小的滑动窗口重复检测，得到重复集合D_2；

7.在集合D_2中检测D_1中的记录的重复记录，得到最终的重复记录集合D；

8.独立地执行步骤 5~9t_{SNM}次，得到t_{SNM}个重复记录集合，并对t_{SNM}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D_F；

BP-OMPN 算法流程图如图 4.5 所示：

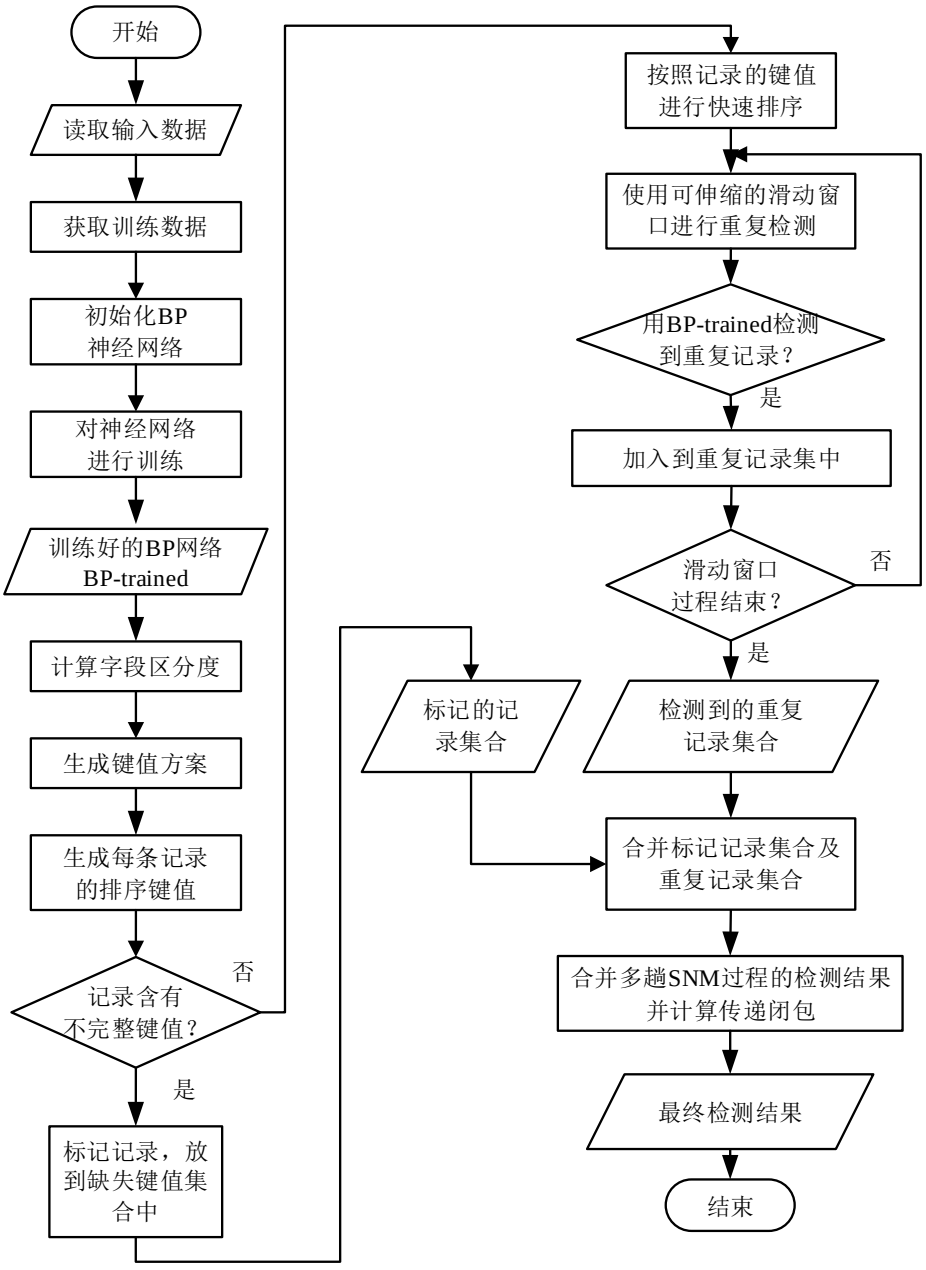

图 4.5 BP-OMPN 算法流程图

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 $\omega_{\max}$ =20，最小滑动窗口大小 $\omega_{\min}$ =3，算法过程中进行单趟 SNM 的次数 t_{SNM} =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 T_{BP}=5000；遗传算法种群初始大小 m_{size}=50，变异概率 p_{muta}=0.05，种群最多迭代 T_{GA}=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。

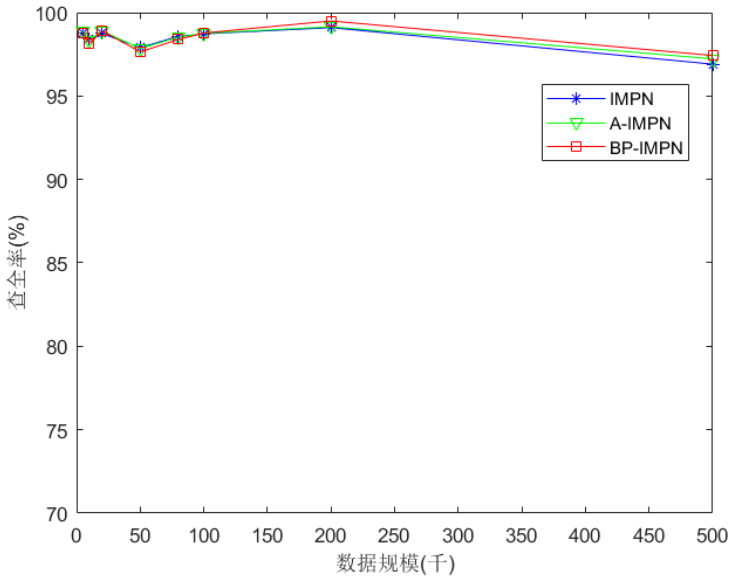

图 4.6 OMPN、A-OMPN、BP-OMPN 算法查全率折线图

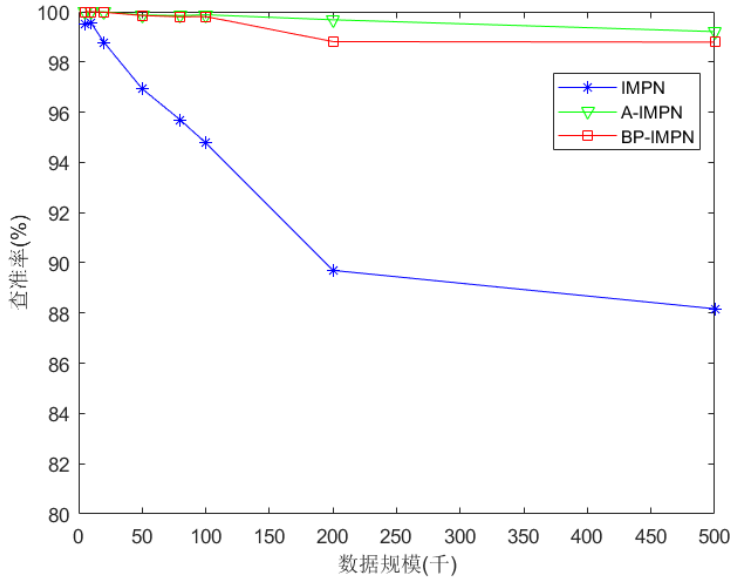

图 4.7 OMPN、A-OMPN、BP-OMPN 算法查准率折线图

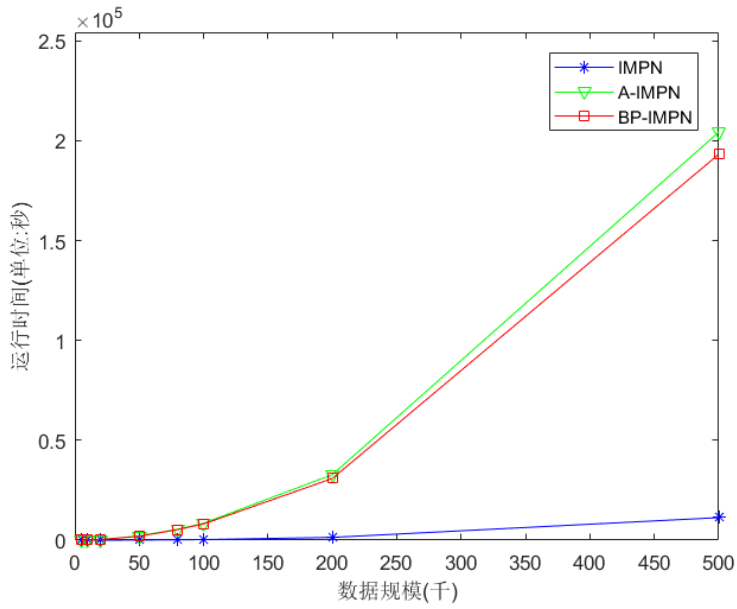

图 4.8 OMPN、A-OMPN、BP-OMPN 算法运行时间折线图

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$O(1)$，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n_I个输入结点、n_H个隐含结点的三层网络，其进行检测的时间复杂度为$O(n_I n_H)$，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[60]，最大训练次数为$T_{BP}=5000$；遗传算法种群初始大小$m_{size}=50$，变异概率$P_{muta}=0.05$，种群最多迭代T_{GA}=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究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”项目的研究。该研究以“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”主要面向研究所内部研究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，包括两种采集模式：人工在线录入以及 Excel 表格批量导入。 （2） 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。 （3） 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。 （4） 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。 （5） 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。 （6） 数据应用模块：在已有数据集的基础上，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≥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：

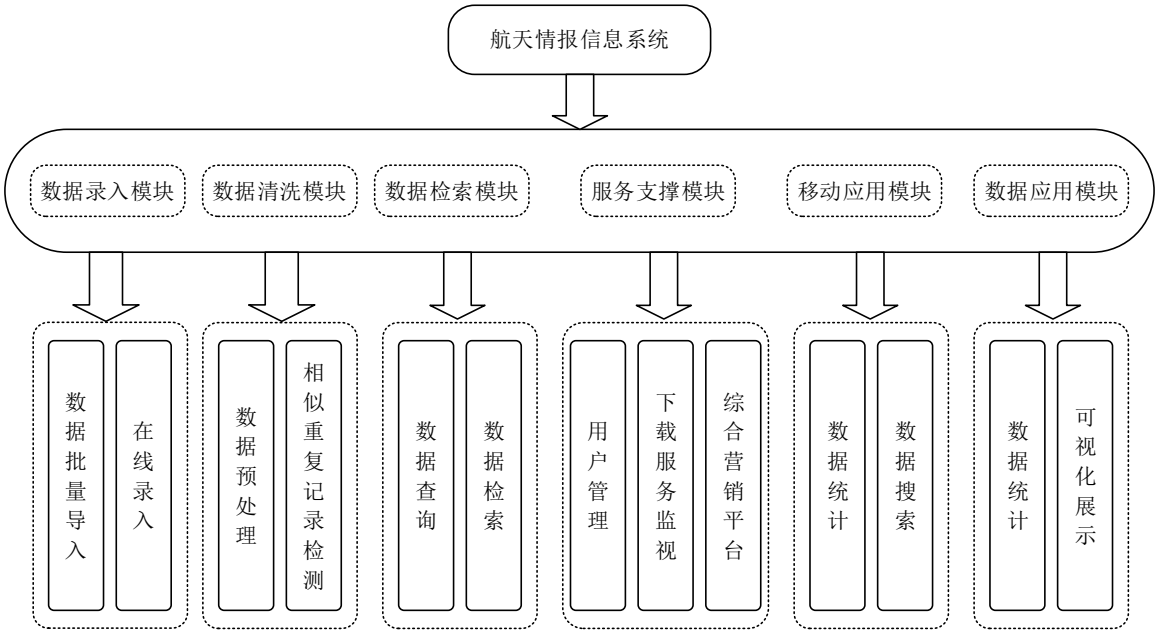

图 5.1 六大功能模块示意图

“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。

前端交互部分主要包括 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：

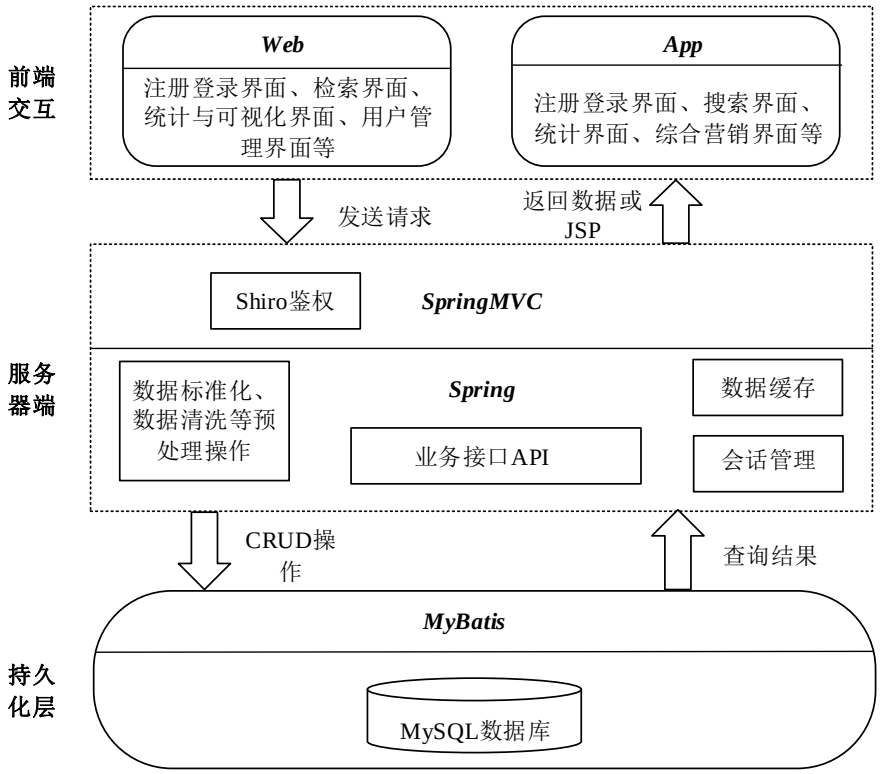

图 5.2 航天情报信息管理系统架构图

“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：

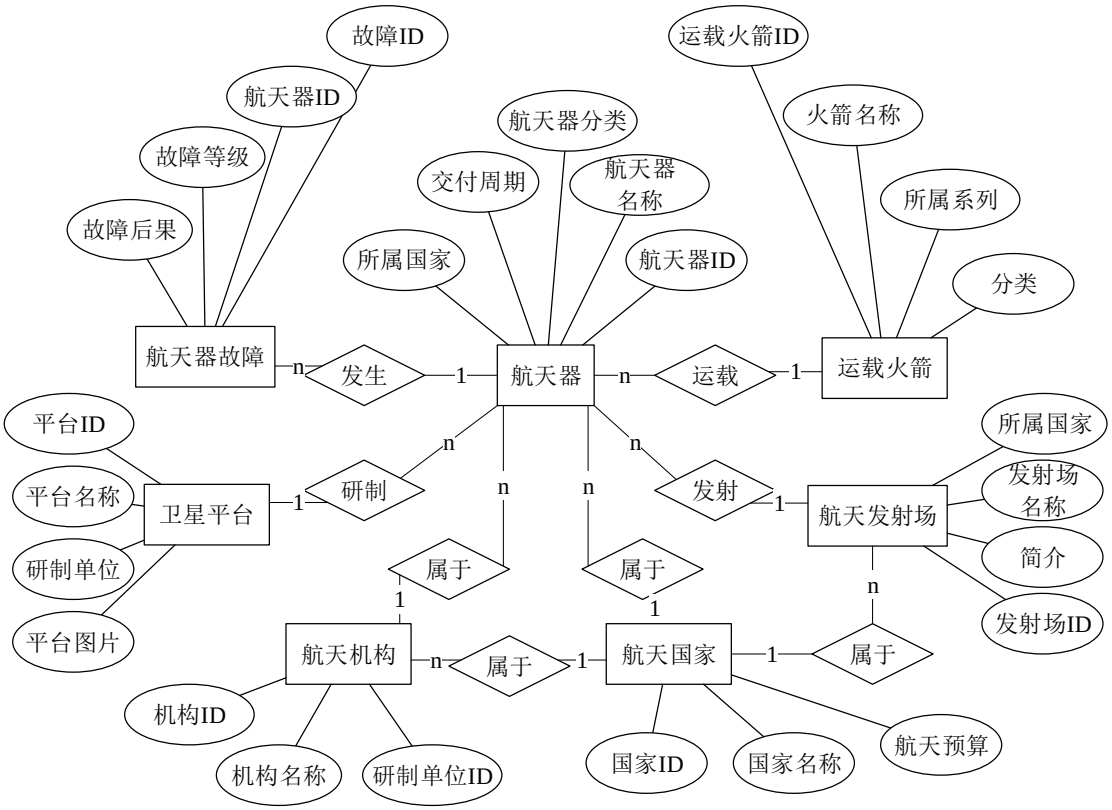

图 5.3 实体关系结构图

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。

（2）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

（3）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
malfunction_id	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	tinyint(1)	否	无
…	…	…	…	…
malfunction_consequence	故障后果	text	是	无

（4）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

<table>
<tr><td rowspan="2"></td><td colspan="2" rowspan="2">服务器端</td><td rowspan="2">Web 端</td><td colspan="2">App</td></tr>
<tr><td>iOS</td><td>Android</td></tr>
<tr><td>开发平台</td><td colspan="2">Windows 7</td><td>Windows 7</td><td>macOS Yosemite</td><td>Windows 7</td></tr>
<tr><td>开发工具</td><td colspan="2">Intellij idea
JDK 1.8.0</td><td>Sublime</td><td>Xcode 8</td><td>Android Studio
JDK 1.8.0</td></tr>
<tr><td>数据库</td><td colspan="2">MySQL</td><td></td><td>SQLite</td><td>SQLite</td></tr>
<tr><td rowspan="2">运行环境</td><td>硬件</td><td>操作系统：Windows server 2008 以上
内存大小：≥4G
磁盘空间：≥300G</td><td rowspan="2">Internet Explorer8 及以上</td><td rowspan="2">iOS 8.0 及以上</td><td rowspan="2">Android 4.0 及以上</td></tr>
<tr><td>软件</td><td>JDK 7.0.71
Tomcat 7.0.54
MySQL 5.6
Navicat 11</td></tr>
</table>

（2）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。

（3）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	请求已发射航天器列表
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

5.3.1 “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究人员维护，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“Lockhead Martin”、“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：

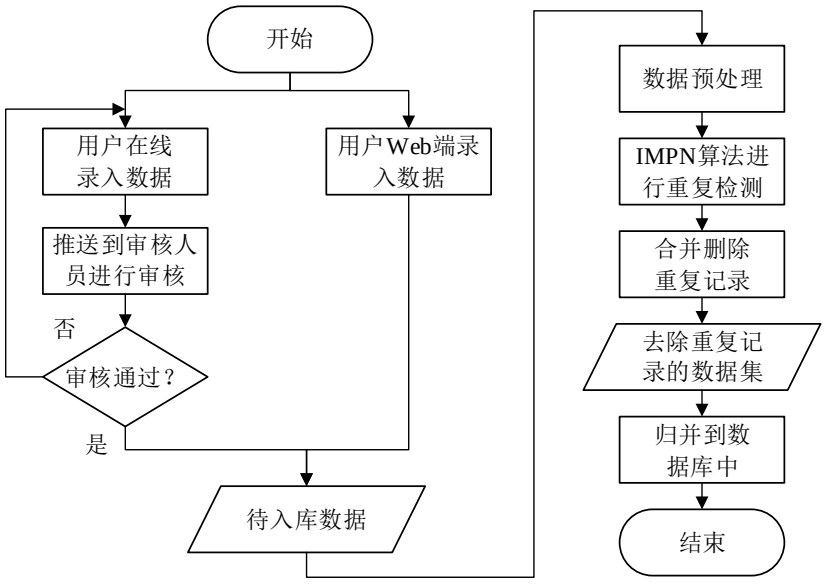

图 5.4 数据清洗业务流程图

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
日期格式检查	日期统一用 yyyy-MM-dd HH:mm:ss 格式
日期非法内容检查	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，因此需要研究出高效的算法，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，本文主要研究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

本文主要研究了数据清洗领域的相似重复记录检测问题，针对研究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究：

（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究。

（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。