一种检测多语言文本相似重复记录的综合方法

A Synthetical Approach for Detecting Approximately Duplicate Database Records of Multi-Language Data

俞荣华　田增平　周傲英

（复旦大学计算机系　上海200433）

Abstract Detecting approximate duplicate records in database is a key problem related to data quality. In this paper, we present a synthetical approach for recognizing clusters of approximately duplicate records of multi-language data. The key ideas are: (1) an efficient algorithm for sorting multi-language data; (2)an efficient edit-distance based pairwise comparison method for multi-language data; (3)using a priority queue of duplicates clusters and representative records strategy to respond adaptively to the data scale.

Keywords Approximate duplicates records, Clustering, Pairwise comparison, Priority queue

1. 前言

随着信息技术的广泛应用，如何有效利用不断激增的数据成为企业的迫切问题。数据仓库和数据挖掘技术为企业从浩瀚的数据海洋中获取有用的知识提供了一种有效的手段。然而，现实世界中的数据往往存在着大量的质量问题，从简单的数据输入错误到相对较复杂的数据间的语义不一致性。如果数据的质量达不到要求，那么数据挖掘这类技术产生的结果也不会理想，甚至产生错误的分析结果，从而误导决策。可见提高数据质量的重要性。

信息重复问题是影响数据质量的关键问题之一。理想情况下，对于现实世界中的一个实体，数据库或数据仓库中应该只有一条对应的记录。根据这样的理解，联邦数据库中的实体标识问题和面向对象数据库中的对象标识问题都可以被看作是信息重复问题。从比较狭隘的观点来看，如果两条记录在某些对应的字段(Field)上的值相等或者足够相近，那么可以认为这两条记录互为重复。我们讨论的相似重复记录检测也以这样的观点为基础。由于这些记录不完全一样，我们称之为相似重复记录。在不致引起混淆的情况下，为了方便，文中有些地方直接称之为重复记录。

检测和去除数据库中的重复记录并非一个新的问题，排序-合并方法是检测数据库中完全重复记录的标准方法[1]。它的基本思想是，先对数据集排序，然后检测相邻记录是否为重复记录。目前已有的检测相似重复记录的方法也大多以此思想为基础，不同之处在于排序对象的选取和记录比较的方法。文[2]中将整条记录视为一个长字符串进行排序，通过计算字符串编辑距离进行记录比较；文[3]以自定义的一个应用相关的健作为排序键，通过一组规则定义的相等理论判定记录是否重复；文[4]以记录的 NGram 值作为排序键，通过计算编辑距离进行记录比较。

上述方法的处理对象都是纯西文表示的数据，不能直接用来处理多语言数据。然而，在使用东方语言的国家里，数据经常是含有多语言的文本。比如下面这样一条描述个人信息的记录：

马明/浙江杭州市凤起路162号 B 座301室/ UT 斯达康通信公司/maming@263.net/6805201.

其中这个记录的有些字段全是中文，有些全是西文，有些既有中文又有西文。

与纯英文数据记录相比，多语言数据记录的排序和记录比较方法都会有所不同，本文中我们以中西文混合情形下的数据记录为例来介绍和讨论我们的方法。本文的主要贡献有：通过序值文件的方式解决了多语言文本的排序问题，并给出了一个有效的排序算法；针对多语言数据，给出了一种合理、高效的记录比较算法，其复杂度为 O(M * N)；采用以聚类为元素的优先队列和代表记录相结合的策略，大大减少了记录比较次数。

本文以中西文混合数据为例介绍了多语言文本排序问题，并给出了有效的排序算法。介绍了多语言数据的记录比较算法，整个检测算法的总体思想以及优先队列等关键策略。给出了实验结果评价和我们的工作。

2. 排序一初步聚类

初步聚类的目标是将潜在的重复记录调整到相邻位置，一般采用排序的方法。对于西文字符而言，排序也就是按字符在字符集中的次序排列(即字典序)。对于汉字而言，存在多种排序方式。在国家标准 GB2312-80中共收集汉字6763个，分成两级。一级字库包括汉字3755个，按拼音字母排序；二级字库包括3008个汉字，按部首排序。由此可见汉字本身的编码不满

俞荣华　硕士研究生，主要研究方向为数据质量和数据清洗。田增平　副教授。周傲英　教授，博士生导师。

19 Chamberlin D, Robie J, Florescu D. Quilt: An XML Query Language for Heterogeneous Data Source. WebDB2000, May 2000

20 Bonifati A, Ceri S. Comparative Analysis of Five XML Query Languages. SIGMOD Record, 2000, 29(1): 68～79

21 Goldman R, Widom J. DataGuides: Enabling Query Formulation and Optimization in Semistructrued Databases. In: Proc. of the 23rd VLDB Conf. Athens Greece, 1997

22 McHugh J, et al. Indexing Semistructured Data: [Technical report]. Stanford University Database Group, 1998

23 McHugh J, Widom J. Query Optimization for XML. In: Proc. of the 25th VLDB Conf. 1999

24 姚建中. XML 文档查询处理与数据库存储的研究:[浙江大学硕士学位论文]. 2000

足任何一种统一的序值规则，不宜作序值用。此外，在多语言的情形下，不同语言字符的序值也需要统一。

2.1 序值文件

我们采取建立序值文件的方式解决序值不统一问题。目前，汉字通常有以下三种排序方式：拼音序、笔划序、部首序。对于汉字各种不同的排序方式，分别建立对应于 GB2312-80 汉字基本集的序值表。序值表中序值的存放按对应的汉字在汉字基本集中出现的顺序进行，因此根据汉字的内码(0XB0A1--0XF7FE)可以直接计算出序值表中存放对应序值的入口地址。计算公式如下：

index=headoffset+[(c1-0xB0H) * 94+ (c2-0xA1H)] * N

其中 c1为汉字内码的第一个字节(区码)；c2为汉字内码的第二个字节(位码)；N 为序值编码的长度，我们用两个字节来存放序值，N=2；headoffset 是序值表中存放第一个汉字(0xB0A1，啊)的位置。对于汉字的任何一种排序方式，西文字符的序值始终在汉字序值之前。我们将汉字序值起始值设定为大于128，这样西文字符编码值在多语言环境下可直接作为序值。

序值表相当于自定义的一种编码，不同的排序方式对应各自的序值表。序值表的大小只有几十 k，可以存放在内存中。根据上述公式，汉字的内码可直接映射为获取序值的地址索引，非常快捷。

2.2 排序算法

一般来说快速排序是所有排序算法中效率最好的，但是对于字符串这样的排序对象，有时候快速排序算法却未必能够获得最佳的效率。在我们的实现方法中，除了传统的快速排序算法外，还采取了一种改进的基数排序算法，该算法能有效地处理不等长多语言文本字符串的排序问题。在实际排序时，根据字符串的长度和记录数在这两种方法中做出选择。

基数排序主要通过分配和收集两个操作来实现。它的最大特点是：(1)时间复杂度为 O(KN)，其中 K 为排序对象的长度，N 为排序对象的个数。因此，有类似于 O(N)的特性，当 K 值较小时算法的效率比较高。(2)基数排序不需要排序对象之间的比较操作，这对字符串对象的排序来说是一大优点，因为两个字符串之间的比较操作开销比较大。

我们取基数 RADIX=256(即一个字节)，因此 K 就等于字段的字节数。由于字符串的长度不相等，为了进一步降低排序的开销，以基数排序为基础，我们采取的算法的基本思想是：所有的排序对象按长度归类，然后按长度从大到小、由后向前进行基数排序的分配收集操作。也就是说对于第 i 个字节的分配和收集操作只针对那些至少有 i 个字节长度的字符串。算法 SortAt 描述如下：

```
输入：数据源：data，排序属性 atrr
输出：排序结果 SVector：array of rid
KeyList：List of (rid, iField);
LenList：array of KeyList;
1. N := ReadFields(data,LenList,attr,K);
2. KeyList := NULL;
3. for i:=K to 0 do
4. begin
5.   LenList[i].tail := KeyList;
6.   KeyList := LenList[i].head;
7.   InitBucket(Bucket);
8.   Distribute(KeyList,Bucket,i);
9.   Collect(KeyList,Bucket);
10. end
11. for j:=0 to N-1 do
12.   SVector[i] := KeyList[i].rid;
13. return SVector;
```

第1行的 ReadFields 函数从数据源 data 中读取排序字段，按字段值长度存放到 LenList 中，所有字符替换成相应的序值，并返回记录个数 N 和字段值最大长度 K。第7行的 InitBucket 将 Bucket 初始化为空。第8行的 Distribute 将 KeyList 中的字段值按第 i 个字节分配到 Bucket 中。第9行的 Collect 将分配到 Bucket 中的字段值按序收集起来，KeyList 指向列表的头部。由上面的算法描述可知，该算法适合于排序对象平均长度较小的情况。在实际使用中我们可以通过比较 K 和 log(N)来决定采用何种排序算法。当 $K>Q*\log(N)$ 时采取快速排序算法，反之采取上述算法。根据我们实验比较的结果，Q 一般要大于2。

3. 记录比较

记录比较的目的是确定两条记录是否为重复记录。判定记录重复可以采取多种方式：文[2]将整条记录视为一个长字符串，然后通过计算字符串编辑距离确定相似与否，该方法过于粗糙；文[3]采用一组规则定义的相等理论来确定记录相似与否。我们采用有效权值模型，通过对某些关键字段的比较来衡量两条记录的相似性。该方法能够更为合理地衡量记录的相似性。对于多语言文本表示的字段，我们可采用分割法细化比较对象。

3.1 基于有效权值的记录相似性度量

不难理解，记录中不同的字段对反映记录特征的贡献不是平均的，比如客户信息中名字显然要比性别更能反应这条记录的特征。基于这样的认识，在衡量两条记录的相似性时，不同的字段应该赋予不同的权重。在造成记录重复的诸多原因中，字段缺失(字段值为空)是其中一个重要原因。如下两条记录：

1. 马明/浙江杭州市凤起路162号 B 座301室/UT 斯达康通信公司/maming@263.net/6805201
2. 马明/浙江杭州市凤起路162号 B 座301室/ /maming@263.net/

相对于记录1，记录2缺少了两个字段的值。如果按逐个字段比较、按权值累计匹配值的方法，这两条记录的相似度将比较低，但事实是它们非常相似。为了消除字段缺失对判断记录相似性的影响，我们引入了有效权值的概念。假定参与比较的字段有 n 个，记录比较 RecordMatch 算法如下：

```
输入：记录 r1，r2
输出：r1，r2的相似度 S
1. validWeight := 0.0;
2. S := 0.0;
3. for i := 1 to n do
4. begin
5.   if(r1.Field[i] == NULL OR r2.Field[i] == NULL ) then
       continue;
6.   validWeight += Weight[i];
7.   S+=Weight[i] * FieldMatch( r1.Field[i], r2.Field[i] );
8. end;
9. if validWeight>0 then S = S / validWeight;
10. return S;
```

Weight[i]为第 i 个属性对应的权重，满足：$\sum$Weight[i] = 1.0。第7行的 FieldMatch 计算两个字段的相似度，在3.2节中详述。在第5行我们可以看到，只有当两条记录在第 i 个字段上对应的值都不为空时，才进行字段比较，对应的权值才计入 validWeight，我们称之为有效权值。在第9行，S 除以 validWeight 得到最终的匹配值 S。考虑到字段缺失是导致重复信息的重要因素之一，采用该方法后有效提高了检测重复记录的准确度。

3.2 字段比较

字段比较是记录比较的基础，文[5]针对西文字符串给出

了若干种字段比较的方法。对于数值这样的非字符串类型的数据，直接进行简单的相等比较。考虑到大部分数据都是字符串类型的，而且重复记录之间的差异往往也是由字符串数据引起的，因此我们重点考虑字符串数据。

字符串相似性比较的典型方法有 Smith-Waterman 算法[6]，其复杂度为 O(m * n * max(m, n))。本文参考了计算编辑距离的思想[7]，采用的算法的复杂度为 O(m * n)。然而，对于多语言数据而言，不同语言的字符之间的比较不仅没有意义，而且会浪费大量的时间。为此我们采取了自然分割的方法，相同语言的字符之间进行比较。考虑如下两个表示地址信息的字符串的比较：

浙江杭州市凤起路/162/号/B/座/301/室

杭州市凤起路/16/号/B/座/301/室

分割之后我们考虑的是：

浙江杭州市凤起路号座室——杭州市凤起路号座室

162B301——16B301

之间的比较。以字段中含有两种字符为例，假定它们在字段值 A 中分别有 m1和 n1个字符，在字段值 B 中分别有 m2和 n2个字符。则在不区分字符类型的情况下，所需比较次数为：

$(m1+n1)*(m2+n2)=m1m2+m1n2+n1m2+n1n2$

在分割之后，所需的比较次数为：$m1m2+n1n2$

以上述例子为例：两种方法的比较次数分别为 $18\times15=270$，$11\times9+7\times6=141$。由此可见，采用分别比较的方法可以大大提高字段比较的效率。

相同类型字符串计算编辑距离的算法 Dist 如下所示：

```
输入：字符串 s1和 s2，以及它们的长度 n1，n2
输出：字符串 s1和 s2的距离
1. float Graph[n1+1][n2+1];
2. Graph[0][0] = 0.0;
3. for col:=1 to n1 do Graph[0][col] = Graph[0][col-1] + In-
     sertCost; //第0行
4. for row:=1 to n2 do Graph[row][0] = Graph[row-1][0] +
     DeleteCost; //第0列
5. for row:=1 to n2 do
6.     for col:=1 to n1 do
7.     begin
8.       d1 :=Graph[row][col-1] + InsertCost;
9.       d2 :=Graph[row-1][col] + DeleteCost;
10.      d3 :=Graph[row-1][col-1] + SubCost(s1[col-1],s2
           [row-1]);
11.      Graph[row][col] := min(d1,d2,d3);
12.    end;
13. return Graph[n1][n2];
```

InsertCost (∈[0,1])定义的是插入一个字符的代价；DeleteCost(∈[0,1])定义的是删除一个字符的代价；第10行是 SubCost(ch1, ch2)计算字符 ch1和 ch2相互替换的代价。当 ch1等于 ch2时，SubCost 返回0。SubCost 可以充分利用输入错误统计信息，在一定程度上减小了输入错误对判断字段相似性的影响。当我们取 InsertCost 等于 DeleteCost 时，上述算法具有对称性，即 s1和 s2的距离也就是 s2和 s1的距离。

算法 Dist 计算的是字符串之间的绝对距离，我们通过以下公式计算相似度：

FieldMatch(s1,s2)=Dist(s1,s2)/MaxDist(s1,s2)

MaxDist(s1,s2)获取字符串 s1、s2之间的最大距离，通常取这两个字符串的最大长度。

4. 重复记录检测算法

第2、3两节分别解决了记录集的初步聚类和记录比较问题，这一节将描述整个检测算法的总体思想，以及其中采取的一些关键策略。

4.1 利用 Union-Find 结构计算传递闭包

我们认为记录的相似重复关系具有传递性，即：如果记录 R1和 R2被判定为相似重复记录，R2和 R3也被判定为相似重复记录，那么我们认为 R1和 R3也是相似重复记录。为了提高检测重复记录的精度，我们采取多轮排序、比较操作。因此在第 m 轮排序、比较操作中检测到的记录的相似重复关系将导致在前 m-1轮中检测到的重复记录聚类的合并，这一过程是一个计算传递闭包的过程。

初始，我们把数据库中的每条记录都视为无向图的一个顶点，每个顶点有如下结构：

struct Node { long rid; int parent; }

rid 表示记录标识，parent ≥ 0时，其值代表父节点位置；parent < 0时，表明该节点为根节点，|parent|表示该连通分量中的顶点个数。当两条记录被判定为重复记录时，将对应顶点所在的连通分量合并。因此，任何时刻无向图中的连通分量就代表了到目前为止检测到的相似重复记录的传递闭包。

根据顶点结构，我们可以看到连通分量以树状结构表示。为了有效解决连通分量的合并操作，我们采用了 Union-Find 数据结构。该结构最早用于不相交集合的并操作[8]，主要有以下两个操作：Union(r1,r2)：将以 r1、r2作为根节点的两棵树合并成一棵新的树，其中节点较少的树成为节点较多的树的根节点的子树，并返回新的根节点。

Find(x)：查找节点 x 所在树的根节点。

初始，我们将每条记录看作是一棵单节点的树，此时每个 Node 的 parent 的值为-1。记录比较过程中，一旦判定两条记录为重复记录，就执行 Union 操作。根据 Union 操作的特点，每一轮中，树的高度最多只会增加一层。因此，第 m 轮中的 Find 操作最多只需回溯 m 次即可找到根节点。

4.2 优先队列与代表记录

传统的重复记录检测方法通过滑动窗口来限定记录比较操作的范围。窗口存放最近扫描过的记录，新进入窗口的记录和窗口中的其余 w-1条记录进行比较操作(w 为窗口的大小)。这种方法对初步聚类的效果比较敏感，因此窗口的大小对检测结果影响较大。我们采取了一种更为有效的优先队列和代表记录相结合的方法。不同于传统的滑动窗口策略，优先队列的每一项代表的是一个重复记录聚类(以对应的树结构的根节点表示)，而不是一条记录。对于一个重复记录聚类而言，单条记录也许并不足以代表整个聚类的特征，因此我们用若干条记录来代表优先队列中的每个聚类的特征。

检测算法根据排序结果顺序扫描数据库。假定当前考察的记录为 R_i，如果 R_i 已经在优先队列的某个聚类中，则将该聚类的优先级设为最高，继续考察下一条记录。否则，R_i 按优先级从高到低和优先队列中各个聚类的代表记录进行比较。假定和 R_i 比较的当前记录为 R_j，一旦判定两者为相似重复记录(相似度高于某一特定值 match_thresh)，则通过 Union 操作合并两者所在的聚类；进一步，如果两者的相似度低于某个 high_thresh(该值大于 match_thresh，但小于1)，说明 R_i 具有一定的代表性，因此将 R_i 作为合并后的聚类的代表记录放入优先队列，这有助于后面检测到更多的潜在重复记录。反之，如果 R_i 和 R_j 的记录比较结果值太低，比如低于某个特定值 low_thresh(该值小于 match_thresh)，说明 R_i 属于 R_j 所在聚类的可能性已经很小，因此无须再和该聚类中的其他代表记录进行比较。此时 R_i 直接和下一个优先级的聚类的代表记录进行比较。

上述过程中，一旦判定 R_i 属于队列中的某个聚类，就不

再进行后续的比较，而是考察数据库中的下一条记录。如果扫描整个优先队列之后，发现 R_i 不属于其中任何一个聚类，则将 R_i 本身所在的聚类加入到优先队列中，并具有最高优先级，同时 R_i 成为该聚类的第一个代表记录。如果此时优先队列的项数已经超出了规定大小，则删除优先级最低的那一项。

以聚类为元素的优先队列结合代表记录，通过适当的设置 high_thresh 和 low_thresh 两个参数，可以大大减少不必要的记录比较操作。不难看出，以上策略几乎不受数据规模的影响，因此我们的方法能够很好地适应数据规模的变化。实验结果证实了这一点。

5. 实验结果

为了全面地考察我们的方法，我们分别用实际数据和测试程序生成的数据进行了测试。评估测试结果的参数有四个：C(Cluster detected)，R(Recall)，P(Precision)，T(Time)。评估参数定义如下(|A|表示集合 A 的元素个数)：

$C=|C_D|/|C_O|$

$R=|R_D|/|R_O|$；

$P=|R_D\cap R_O|/|R_D|$；

T 为算法执行的时间

其中，R_O 为测试数据中实际的重复记录集合；R_D 为系统检测到的重复记录集合；C_O 为测试数据包含的聚类数目；C_D 为检测到的聚类数目。本文实验在 PC 工作站上完成，工作站的硬件配置为：CPU PentiumII350，RAM 128M；操作系统为 Windows2000 Professional。

5.1 实际数据

实际数据来自某网站的关于同学录信息的数据。由于每个人可以加入到初中、高中、大学等各个时期的班级的同学录中，许多人还加入了同时期相邻班级的同学录中。而在每次加入一个新的同学录时该网站都要求重新输入个人信息，从而导致了大量重复记录的存在。我们获取了某一地区各类班级的大约5000条记录。其中 $|C_O|=1253$，$|R_O|=3178$。这两个数据是在我们的系统对实际数据分批检测的结果和人工检视相结合的基础上得到的，因此可以大致反应实际数据的重复情况。测试结果如下：

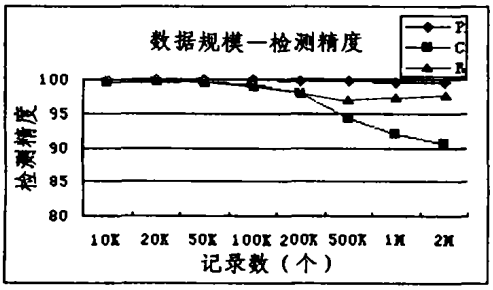
$|C_D|=1245$，$|R_D|=3147$，$|R_D\cap R_O|=3132$，$T=3.28s$

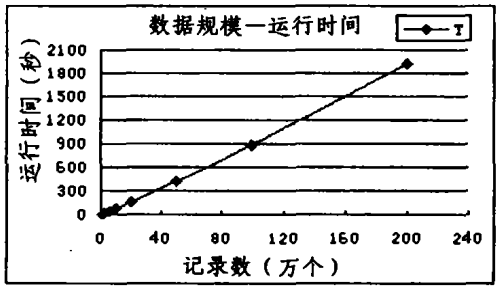
则：$C=99.4\%$，$R=99.02\%$，$P=99.5\%$

5.2 模拟数据

由于实际数据的规模有限，很难作全面的测试。我们构造了能够生成不同质量和数据规模的测试数据生成程序。生成测试数据的主要参数有：S_SIZE：样本数据规模；P_D：重复率；M_D：最大重复数；EP_E：字段为空的概率；EP_T：字段输入错误的概率；EP_R：单个字段中，出错部分允许的最大百分比。

在不同数据规模下，实验结果如以下两图所示：





在左边这张图中，我们可以看到随着数据量的增长，各个检测精度指标均有所下降。但是，即便在200万条记录的规模下，各项指标仍然能够达到90%以上。而 P 曲线始终维持高百分比(99%以上)。在运行时间这张图中，我们可以看到随着记录数的增长，运行时间也相应增长，但两者呈线性关系，因此我们的方法总的复杂度为 O(N)。

通过实验，我们可以看到本文方法的显著优点是：(1)总复杂度仅为 O(N)；(2)处理大量数据的能力强；(3)检测的精度高。

结束语 信息重复问题是影响数据质量的关键问题。本文研究了在多语言文本条件下如何检测相似重复记录的问题。提出了一种有效的综合方法。该方法的优点是时间复杂度小，检测精度高，能很好地适应数据规模的变化。接下去的工作是将这部分功能整合到我们的数据清洗工具包中，使其成为提高数据质量的一个实用工具。

参 考 文 献

1 Bitton D，DeWitt D J. Duplicate record elimination in large data files. ACM Transactions on Database Systems，1983，8(2)：255～265

2 Monge A E，Elkan C P. An efficient domain-independent algorithm for detecting approximately duplicate database records. 1997

3 Hernandez M，Stolfo S. The merge/purge problem for large databases. In：Proc. of the ACM SIGMOD International Conference on Management of Data，May 1995. 127～138

4 邱越峰，田增平，季文赟，周傲英. 一种高效的检测相似重复记录的方法. 计算机学报，2001，24(1)：69～77

5 Monge A E，Elkan C P. The field matching problem：Algorithms and applications. In：Proc. of the 2nd Int. Conf. on Knowledge Discovery and Data Mining，1996. 267～270

6 Smith T F，Waterman M S. Identification of common molecular subsequences. Journal of Molecular Bilogy，1981，147：195～197

7 Lowrance R，Wagner R A. An extension of the string-to-string correction problem. J. ACM，1975，22(2)：177～183

8 Tarjian R E. Effiency of a good but not linear set union algorithm. Journal of the ACM，1975，22(2)：215～225