An Important Issue in Data Mining-Data Cleaning

Qi Xiao Yang
Institute of High Performance of Computing
89B Science Park Drive#01-05/08 the Rutherford
Singapore 118261
qixy@ihpc.nus.edu.sg
tel: (65)7709265

Sung Sam Yuan, LuChun
School of Computing
National University of Singapore
3 Science Drive 2, Singapore 117543
{ssung,luchun}@comp.nus.edu.sg
tel: (65)8746148

Jay Rajasekera
Graduate School of International Management
International University of Japan
jrr@iuj.ac.jp
tel: (81) 257791531

1. Introduction

Data mining techniques are well accepted for discovering potentially useful knowledge from the large datasets. Our past research work on studying aspects of data mining includes improving the performance of the rule generation [SS95, SWC96], extending the scope of association rule mining [RSC99] and data generation [SLL96].

Prior to data mining process, data cleaning is essential in that the quality of rules derived from the mining process is subject to the quality of data. Recently, significant attention is paid to record de-duplication----an important branch of data cleaning. Various reasons are behind different representations of identical record: typographical errors, purposeful entry of false names, inconsistent data formats, incomplete information and registrant moving from one place to another.

[LL+99] is a milestone paper in the area of record de-duplication. Experiments on real-world data demonstrate that the methods of de-duplicate records presented in [LL+99] are efficient. Further study indicates that there is still room for improvement in the core part of its whole technology---- the algorithm of the calculation of the Field Similarity. Our paper is to introduce a new algorithm to calculate Field Similarity. Theoretical analysis, concrete examples and experimental result shows that our algorithm can significantly improve the accuracy of the calculation of Field Similarity.

The rest of the paper is organized as follows. Section 2 gives a background description of the algorithm of calculating Field Similarity presented in [LL+99]. Section 3 proposes our algorithm of calculating Field Similarity and exhaustively compares the new algorithm with the previous one. Section 4 provides an experiment to prove the performance improvement with the introduction of the new algorithm.

2. Preliminary Background

This section gives a brief description of the algorithm to calculate Field Similarity presented in [LL+99].

Let a field in record X have words Ox1, Ox2,….., Oxn and the corresponding field in record Y have words Oy1, Oy2,……, Oym. Each word Oxi ,1 $\leq$ i $\leq$ n is compared with words Oyj, 1 $\leq$ j $\leq$ n. let DoSx1, DoSx2,….., DoSxn, DoSy1, DoSy2,….., DoSym be the maximum of the degree of similarities for words Ox1, Ox2,….., Oxn, Oy1, Oy2,……, Oym respectively. Then the Field Similarity for record X and Y

$$\text{SIMF(X,Y)} = \frac{\sum_{i=1}^{n} DoS_{x_i} + \sum_{j=1}^{m} DoS_{y_j}}{n+m} \qquad (1)$$

About the calculation of degree of similarity of words---DoS:

- If two words are exactly the same, the degree of similarity between these two words is 1.

- If there is a total of x characters in the word, then we deduct $\frac{1}{x}$ from the maximum degree of similarity of 1 for each character that is not found in the other word. For example, if we compare "kit" and "quit", then DoS $_{kit}$=1-$\frac{1}{3}$=0.67 since the character k in "kit" is not found in "quit" and DoS $_{quit}$=1-$\frac{2}{4}$=0.5 since the characters q and u in "quit" are not found in "kit".

Exercise: compute the Field Similarity of the field "address" of record 1and 2 in table 1.

Record	Name	Address
1	Qi Xiao Yang	129 Industry Park
2	Qi Xiao Yang	129 Indisttry Park

Table 1 exercise for calculation of degree of similarity of words

1. The degree of similarity between "129" and "129" is 1, between "129" and "Indisttry" is 0, between "129" and "Park " is 0. So according to the above rule, DoS129 $_{R1}$ =1. (***DoS should be the maximum***)
2. The degree of similarity between "Industry" and "129" is 0, between "Industry" and "Indisttry" is 1-$\frac{1}{8}$=0.875, between "Industry" and "Park " is 0. So according to the above rule, DoSIndustry $_{R1}$ =0.875.
3. In the same way, we will obtain the following:

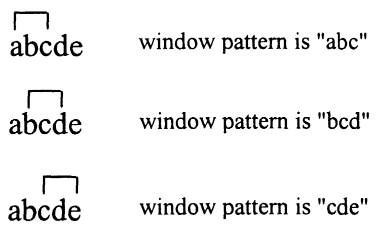
   DoSPark $_{R1}$ =1, DoS129 $_{R2}$ =1, DoSIndisttry $_{R2}$ =1-$\frac{2}{9}$=0.778, DoSPark $_{R2}$ =1
4. When Formula 1 is employed, the address Field Similarity for R1 and R2 can be obtained as:

$$\mathrm{SIMF(X,Y)} = \frac{\sum_{i=1}^{n} DoS_{x_i} + \sum_{j=1}^{m} DoS_{y_j}}{n+m} = \frac{1+0.875+1+1+0.778+1}{6} = 0.942$$

3. Proposed New Algorithm of Calculation of Field Similarity

This section proposes a new algorithm----Moving Contracting Window Pattern Algorithm (MCWPA) to calculate Field Similarity.

Firstly, we give the definition of **window pattern.** All characters as a whole within the window constitute a **window pattern**. Take a string "abcde" as an example, when the window is sliding from the left to the right with the window size being 3, the series of window patterns obtained are "abc", "bcd" and "cde".



Let a field in record X have n characters (including blank space or comma, this applies to the following) and the corresponding field in record Y have m characters. *w* represents window size, Fx represents the field of record X and Fy represents the field of record Y. The Field Similarity for record X and Y is

$$\mathrm{SIMF(X,Y)}=\sqrt{\frac{SSNC}{(n+m)^2}} \qquad (2)$$

SSNC represents the Sum of the Square of the Number of the same Characters between Fx and Fy. SIMF(X,Y) approximately reflects the ratio of the total number of the common characters in two fields to the total number of characters in two fields.

The following algorithm (MCWPA) is to calculate *SSNC.*

```
1.  w= the smaller of n and m;
2.  SSNC=0;
3.  Fs=the smaller of Fx and Fy;
4.  window is placed on the leftmost position;

5.  while ((window size is not 0) or (still some characters in Fs are accessible))
6.  {
7.      while (window right border does not exceed the right border of the Fs )
8.      {
9.          if ( the window pattern in Fx has the same pattern anywhere in Fy )
10.         {
11.             SSNC= SSNC +$(2w)^2$ ;
12.             mark the pattern characters in Fx and Fy as inaccessible characters to
    avoid revisiting;
13.         }
14.         move window rightward by 1 (if the window left border is on an inaccessible character,
    move window rightward by 2 and so on and so forth)
15.     }
16.     w=w-1;
17.     window is placed on the leftmost position where the window left border is on an
    accessible character;
18. }
19. return SSNC;
```

Figure 1 MCWPA algorithm

Several examples are provided to illustrate how to calculate the Field Similarity with MCWPA and formula (2).

Example 1: calculate the following Field Similarity.

Field 1	abcd
Field 2	abcd

In this example, n=4,m=4, Fx =Abcd, Fy=Abcd, the initial value for *w* is 4, that means, initially the window is placed in the leftmost position with the window size 4 and the window pattern is "Abcd". When line 9 is executed in the program (figure 1), it finds the same pattern "Abcd" in Fy. When line 11 is executed, *SSNC* changes to $(2*4)^2 = (8)^2$. When line 12 is executed, "Abcd" in Fx and "Abcd" in Fy are all marked as inaccessible characters. After lines 13, 14, 15,16, 17 are executed sequentially, the program runs back to line 5. Because "Abcd" are all marked as inaccessible characters in Fs, the condition "still some characters in Fs are accessible" is false. The program ends. According to formula (2)

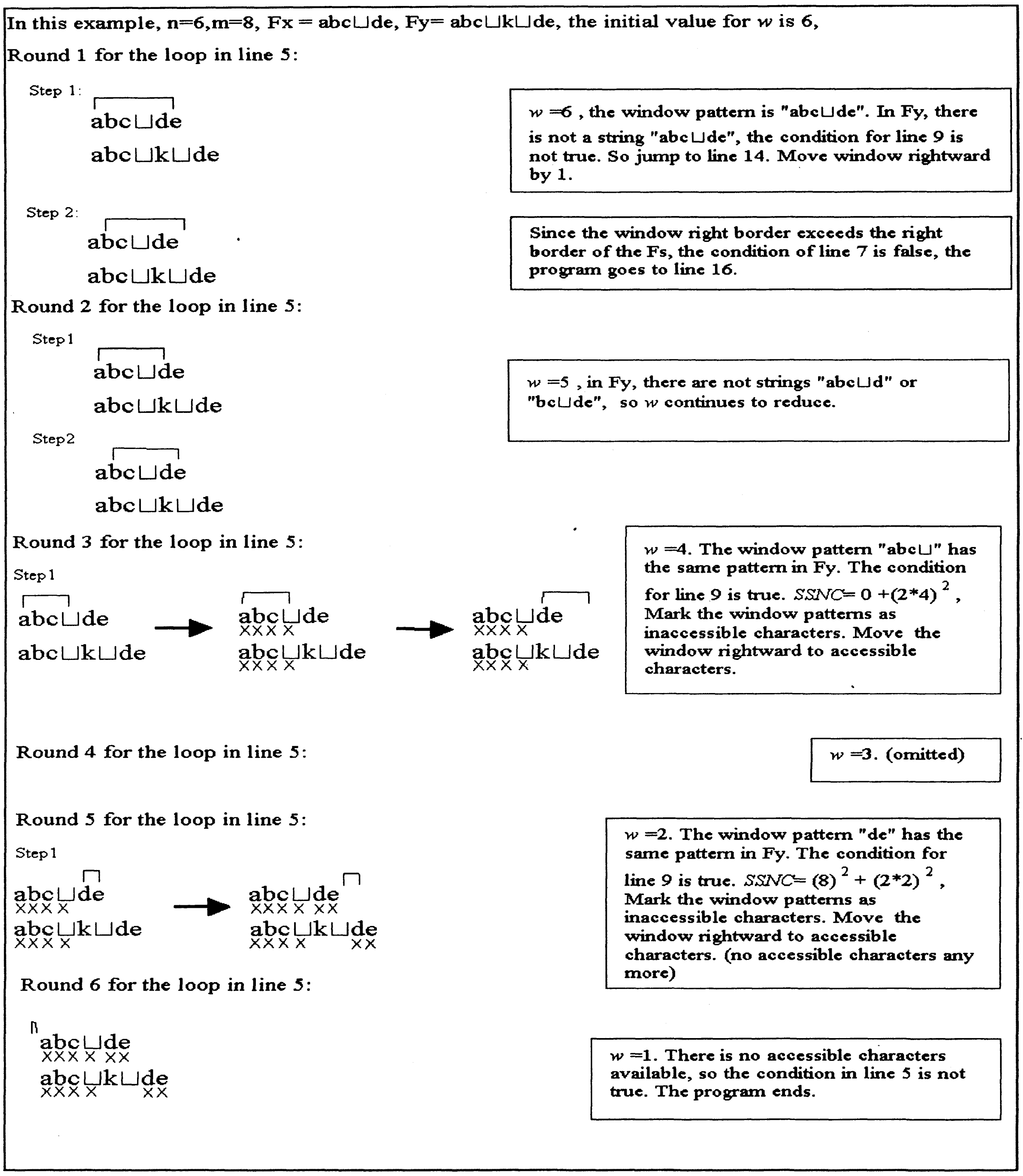
$$\mathrm{SIMF(X,Y)}=\sqrt{\frac{SSNC}{(n+m)^2}}=\sqrt{\frac{8^2}{(4+4)^2}}=100\%$$

So if two fields are exactly the same, SIMF(X,Y) is 100%.

Example 2: calculate the following Field Similarity.

Field 1	abc de
Field 2	abc k de

The process of calculating SSNC with MCWPA is shown as follows.



In this example, n=6,m=8, Fx = abc⊔de, Fy= abc⊔k⊔de, the initial value for *w* is 6,

Round 1 for the loop in line 5:

Step 1:

abc⊔de

abc⊔k⊔de

***w* =6 , the window pattern is "abc⊔de". In Fy, there is not a string "abc⊔de", the condition for line 9 is not true. So jump to line 14. Move window rightward by 1.**

Step 2:

abc⊔de

abc⊔k⊔de

Since the window right border exceeds the right border of the Fs, the condition of line 7 is false, the program goes to line 16.

Round 2 for the loop in line 5:

Step1

abc⊔de

abc⊔k⊔de

***w* =5 , in Fy, there are not strings "abc⊔d" or "bc⊔de", so *w* continues to reduce.**

Step2

abc⊔de

abc⊔k⊔de

Round 3 for the loop in line 5:

Step1

abc⊔de → abc⊔de → abc⊔de

abc⊔k⊔de → abc⊔k⊔de → abc⊔k⊔de

***w* =4. The window pattern "abc⊔" has the same pattern in Fy. The condition for line 9 is true. *SSNC*= 0 +$(2*4)^2$, Mark the window patterns as inaccessible characters. Move the window rightward to accessible characters.**

Round 4 for the loop in line 5:

***w* =3. (omitted)**

Round 5 for the loop in line 5:

Step1

abc⊔de → abc⊔de

abc⊔k⊔de → abc⊔k⊔de

***w* =2. The window pattern "de" has the same pattern in Fy. The condition for line 9 is true. *SSNC*= $(8)^2$ + $(2*2)^2$, Mark the window patterns as inaccessible characters. Move the window rightward to accessible characters. (no accessible characters any more)**

Round 6 for the loop in line 5:

abc⊔de

abc⊔k⊔de

***w* =1. There is no accessible characters available, so the condition in line 5 is not true. The program ends.**

According to formula 2,

$$SIMF(X,Y) = \sqrt{\frac{SSNC}{(n+m)^2}} = \sqrt{\frac{(2*4)^2+(2*2)^2}{(6+8)^2}} \approx 63\%$$

The introductory part mentions that SIMF(X,Y) approximately reflects the ratio of the total number of the common characters in two fields to the total number of characters in two fields. For this example, There are altogether 2*6=12 common characters ("abc ", "de") in 14 characters ("abc de" and "abc k de"), so SIMF(X,Y) should be $\frac{12}{14}\approx 85\%$, why is the result equal to 63%? Because these 6 characters are not continuous. Example 5 will give a detailed discussion about this issue.

3.1 Analysis and Comparison of Two Algorithms of Field Similarity

This section will give some examples to show that MCWPA can overcome some drawbacks that exist in the previous algorithm of the Field Similarity. Also the logic behind the design of MCWPA is presented.

Example 3: calculate the following Field Similarity with the above two algorithms.

Field 1	ex ex ex ex ex ex ex ex ex ex
Field 2	ab ab ab ab ab ab ab ab ab ex

With the previous algorithm,

$$\text{SIMF(X,Y)} = \frac{\sum_{i=1}^{n} DoS_{x_i} + \sum_{j=1}^{m} DoS_{y_j}}{n+m} = \frac{11}{20} > 50\%$$

With MCWPA,

$$\text{SIMF(X,Y)} = \sqrt{\frac{SSNC}{(n+m)^2}} = \sqrt{\frac{(2*3)^2}{(2*29)^2}} = \frac{3}{29} \approx 10\%$$

Obviously, the two fields are quite different, only 10% common characters. However, the result of the previous algorithm shows that these two fields have 50% similarity. In contrast, the result of MCWPA is about 10%, which is quite close to the expectation.

Analysis: This example shows that there is a drawback for the previous algorithm. In it, DoSx1, DoSx2,....., DoSxn, DoSy1, DoSy2,....., DoSym are the **maximum** of the degree of similarities for words Ox1, Ox2,....., Oxn, Oy1, Oy2,......, Oym respectively. If quite a number of words in one field are similar to only one word in the other field and dissimilar to other words, the previous algorithm will give inaccurate result. MCWPA overcomes this problem by marking the same characters in two fields as inaccessible so as to avoid revisiting.

Example 4: calculate the following Field Similarity for two cases with the above two algorithms.

Case1:

Field 1	de abc
Field 2	de abc

Case2:

Field 1	abc de
Field 2	de abc

With the previous algorithm for case 1: SIMF(X,Y) =1,
for case 2: SIMF(X,Y) =1

With MCWPA for case 1: $\text{SIMF(X,Y)} = \sqrt{\frac{SSNC}{(n+m)^2}} = \sqrt{\frac{(2*6)^2}{(6+6)^2}} = 1$,

for case 2: $\text{SIMF(X,Y)} = \sqrt{\frac{SSNC}{(n+m)^2}} = \sqrt{\frac{(2*3)^2+(2*2)^2}{(6+6)^2}}$

$=0.6 \approx 60\%$

Note: for case1, two algorithms produce the same result.

Analysis: Clearly, the similarity in case 1 should be higher than that in case 2. However, the same results based on the previous algorithm suggest that the previous algorithm considers "abc de" and "de abc" in case 2 the same. This disagrees with our common sense. In the following experiment section, we will show that this is fatally erroneous in some dataset with Chinese names. Further study of the previous algorithm shows that the adoption of word as basic unit results in its inability to distinguish between two exactly the same fields and two fields with the same words in different sequences. To improve the accuracy, MCWPA uses the character as the unit. In this example, if the unit is word, both case 1 and case 2 have two same words. In contrast, if the unit is character, case 1 has 6 same characters and case 2 has 5 same characters. As expected, SIMF(X,Y) in case 1 is larger than SIMF(X,Y) in case 2 when MCWPA is employed.

Example 5: calculate the following Field Similarity for two cases with the above two algorithms.
Case1:

Field 1	Fu Hui
Field 2	Mr Fu Hui

Case2:

Field 1	Fu Hui
Field 2	Fu Mr Hui

With the previous algorithm for case 1: SIMF(X,Y) =80%,
for case 2: SIMF(X,Y) =80%,

With MCWPA for case 1: $$\text{SIMF(X,Y)} = \sqrt{\frac{SSNC}{(n+m)^2}} = \sqrt{\frac{(2*6)^2}{(6+9)^2}} = 80\%,$$

for case 2: $$\text{SIMF(X,Y)} = \sqrt{\frac{SSNC}{(n+m)^2}} = \sqrt{\frac{(2*2)^2+(2*4)^2}{(6+9)^2}} \approx 60\%$$

Note: for case1, two algorithms produce the same result.

Analysis: Intuitively, in case 1, "Fu Hui" and "Mr Fu Hui" should be the same person. In case 2, the likelihood exists that due to transposition error, originally "Fu Mr Hui" should be " Mr Fu Hui". However, in more likelihood, due to typographical errors, originally "Fu Mr Hui" should be " Fu Mi Hui" or "Fu Ma Hui", etc. Factually, the two common words "Fu Hui" in field 2 of case 1 are continuous. In contrast, in field 2 of case 2, they are interpolated by another word "Mr", hence the similarity between two fields is severely reduced. Thus intuitively and factually two fields in case 1 should be more similar than those in case 2. However, the previous algorithm gives the same results for case 1 and case 2. In contrast, the results based on MCWPA show that the similarity for case 1 is reasonably higher than that for case 2. With respect to characters, both case 1 and case 2 have 6 common characters ("Fu" " Hui"). According to example 4, even MCWPA can not distinguish case1 from case 2. Further examination of the two cases reveals that in field 2 of case 1, these 6 characters are continuous while in field 2 of case 2, they are not. In order to reflect the difference in terms of continuity despite the same number of common characters, MCWPA introduces the square to the calculation of SIMF(X,Y). In the calculation of SIMF(X,Y) in example 5 with MCWPA, the fundamental reason that the result of case1 is larger than that of case2 is because $6^2 > 2^2 + 4^2$. Mathematically, it is easily seen that the square of the sum of numbers is larger than the sum of the square of numbers, that is, $(a+b+....+n)^2 > a^2+b^2+......+n^2$, (if $a \neq b..... \neq n \neq 0$). In this way, the introduction of square in the calculation of SIMF(X,Y) can overcome the continuity problem which leads to the inaccurate result for the previous algorithm. Now, the answer to the remaining question in example 2 is obvious, why is the result equal to 63% that is much lower than $\frac{12}{14}$? Because those 6 common characters in example 2 are not continuous. If the common characters in field 2 for the example 2 are continuous like the following:

Field 1	abc de
Field 2	abc de k

The result will be $\frac{12}{14}$. So MCWPA correctly reflects the discontinuity by reducing reasonable amount from the result.

3.2 The Comparison of Time Complexity between two Algorithms

For pedagogical reasons, suppose we have two fields with the same number of words (W) and same number of characters (N).

For the previous algorithm:

Complexity of calculation of Field Similarity by the previous algorithm is $O(W^2)$, no matter it is **worst case** or **average case** because every word in one field needs to be compared with every word in the other field to find the maximum DoS.

For MCWPA:

Complexity for the ***worst case*** (no common characters exist between two fields):

When the window size is N, the complexity is $O(1^2)$.

When the window size is N-1, the complexity is $O(2^2)$.

$\vdots$

When the window size is 1, the complexity is $O(N^2)$.

The total is:

$$\overbrace{1^2+2^2 \ldots\ldots +N^2}^{N} = \frac{1}{6}N(N+1)(2N+1)$$

So the complexity for the worst case of calculation of Field Similarity by MCWPA is: $O(N^3)$.

It is not true to say that time complexity is the disadvantage of MCWPA since $O(N^3)$ only applies to the worst case. Through the following two examples, we prove that by some reasonable omission, MCWPA can be more efficient than the previous algorithm.

Example 6: compare the number of operations involved in the calculation of Field Similarity by two algorithms.

Field 1	ab cd ef gh ij kl mn op qr st uv wx yz
Field 2	ab cd ef gh ij kl mn op qr st uv wx yz

With the previous algorithm:

Since every word in field 1 needs to be compared with every word in field 2 to find the maximum DoS and both fields have 13 words, the total number of operations is 13^2. (We omit the operations irrelevant to the current discussion)

With MCWPA:

The whole program ends at the first loop when the window size is equal to the length of Field 1 or Field 2. So the number of operations is 1.

Analysis: this example shows that under some circumstances, MCWPA does take less time than the previous one. The more the two fields are similar, the less time it takes.

Example 7: compare the number of operations involved in the calculation of Field Similarity by two algorithms.

Field 1	ab cd ef gh i n k l
Field 2	ab cd ef gh i m o l

With the previous algorithm:

Like the example 6, the total number of operations is 8^2 .

With MCWPA:

The numbers of operations for the window size 19, 18, 17, 16, 15 are 1^2 , 2^2 , 3^2 , 4^2 , 5^2 ,respectively. When the window size is 14, it finds the first matching pattern "ab cd ef gh i⊔". After the strings "ab cd ef gh i⊔" are marked as inaccessible in two fields, the window size becomes 5, 4, 3, 2, the corresponding numbers of operations are 1^2 , 2^2 , 3^2 , 4^2 , respectively. When the window size is 2, it finds another matching pattern "⊔l". Likewise, the numbers of the remaining operations are 1^2 , 2^2 , 3^2 . So the total number of operations is:

$(1^2 + 2^2 + 3^2 + 4^2 + 5^2) + (1^2 + 2^2 + 3^2 + 4^2) + (1^2 + 2^2 + 3^2)$

$=55+30+14$

$=99$

Analysis: It can be seen that to find all matching patterns, MCWPA uses 99 operations with SIMF(X,Y) being 0.692 while the previous algorithm only uses 64 operations. However, to find the first pattern "ab cd ef gh i⊔", it uses only $1^2 + 2^2 + 3^2 + 4^2 + 5^2 = 55$ operations. Moreover, the contribution of the first pattern to SSNC is quite substantial, $(2*13)^2=676$. (The length of the string "ab cd ef gh i⊔" is 13) In comparison, to find the second pattern "⊔l " with the contribution of only $(2*2)^2=16$ to SSNC, it uses as many as $1^2 + 2^2 + 3^2 + 4^2 = 30$ operations. The characteristic of MCWPA is that it picks out the longer matching strings (with more contribution to SSNC) at earlier stage. This gives rise to the possibility of improving the efficiency of MCWPA by introducing user-specified execution-termination window size threshold. For example, in example 7, if the execution-termination threshold is half of the window size, after the program executes $1^2 + 2^2 + 3^2 + 4^2 + 5^2 = 55$ operations and finds the matching pattern "ab cd ef gh i⊔", it will end with SIMF(X,Y) being 0.684 because the window size in the next step is 5 which is less than the execution-termination threshold. There are two reasons for the introduction of the execution-termination window size threshold:

1) The contribution of short matching patterns is insignificant to the calculation of SSNC. For this example, the difference in SIMF(X,Y) is only 0.692-0.684=0.008. In fact, even though two fields have some short matching patterns, there is no reason to believe that the two fields are similar. For example, although "ab cd ef" and "fa ce db" have 6 1-character-long matching patterns, ("a","b","c","d","e","f") intuitively, they are totally dissimilar.
2) Based on the above analysis of time complexity of MCWPA:

   When the window size is N, the complexity is $O(1^2)$.

   When the window size is N-1, the complexity is $O(2^2)$.

   ⋮

   When the window size is 1, the complexity is $O(N^2)$.

It can be seen that the smaller the window size is, the higher the time complexity is. So the introduction of the execution-termination threshold, that is, the omission of the calculation of small window size, can reduce time complexity of MCWPA significantly.

Thus, with the introduction of the suitable execution-termination window size threshold, MCWPA can be more efficient than the previous algorithm. Of course, there is a tradeoff between faster execution time and more accurate result.

4 Experiment Result

The dataset used to compare two algorithms is a merger of two datasets that come from two surveys conducted through an electronic form within a mass-sent email. The dataset has 782 records. Each record contains seven fields, namely, Matric Number, Name, Address, Major, Grade, Race, Gender. Manual inspection shows 50 duplicate records. Various problems include matching records with different Matric number or mis-spelled names.

To detect duplicate records, computation of record similarity needs to be carried out which, in turn, requires the knowledge of field weightage---the importance of each field. For example, the name field should have higher weightage than sex field because name is more informative than sex. The sum of all field weightage should be 1.

Let a database has data fields F1, F2,...., Fn with field weightages W1, W2,....., Wn respectively. Given two records X and Y, let SIMF1(X,Y), SIMF2(X,Y),....., SIMFn(X,Y) be the field similarities for F1, F2,...., Fn respectively, then the record similarity for record X and Y is $\sum_{i=1}^{n} \mathrm{SIM}_{\mathrm{F}i}(X, Y) * Wi$

Among seven fields, the most distinguishing fields--- Name, Address and Matric number are chosen for the calculation of Field Similarity. We compare two algorithms by two criteria: 1) Miss Detection (duplicate records are not detected) and 2)False Detection (similar non-duplicate records are treated as duplicate records)

1) Miss Detection

- The previous algorithm failed to detect 5 pairs of duplicate records. That is, it has 5/50=10% Miss Detection rate.
- MCWPA also failed to detect 5 pairs of duplicate records. That is, it has 5/50=10% Miss Detection rate.

2) False Detection

- The previous algorithm incorrectly matched **10 pairs** of non-duplicate records.
- MCWPA incorrectly matched **only 1 pair** of non-duplicate records.

Analysis:

The results show that with respect to Miss Detection, two algorithms have roughly the same performance. However, in terms of False Detection, MCWPA performs much better than the previous algorithm. Further study of the testing dataset shows that in the name field, there are some similar non-duplicate Chinese names such as 高华明 and 高明华. These names are stored as English characters in the dataset, namely, "Gao Hua Ming" and "Gao Ming Hua". As analyzed in the example 4, the previous algorithm treats two fields with the same words in different sequences as the matching fields. Thus the high False Detection rate for the previous algorithm begins to make some sense. In addition, there is also a case that the previous algorithm treats "zeng hong" and "zeng zeng" the same. As analyzed in the example 3, MCWPA identifies a large difference in the calculation of Field Similarity between these two fields.

Conclusion

This paper has presented a new algorithm (MCWPM) for the calculation of Field Similarity. After a background description of the algorithm of calculating Field Similarity presented in [LL+99] is given, our new algorithm (MCWPM) of calculating Field Similarity is proposed. In essence, MCWPM improves the previous algorithm in the following aspects:

1) The introduction of marking the common characters as inaccessible to avoid revisiting, which is presented in example 3.
2) The adoption of the character as unit for the calculation of Field Similarity instead of words to improve accuracy, which is presented in example 4.
3) The introduction of square to the calculation of Field Similarity to reflect the difference in terms of continuity despite the same number of common characters, which is presented in example 5.
4) The introduction of execution-termination window size threshold to achieve higher efficiency, which is presented in example 6 and 7.

Thorough comparison of these two algorithms through theoretical analysis, concrete examples and experimental result leads to the conclusion that our new algorithm (MCWPM) can significantly improve the accuracy of the calculation of Field Similarity.

References

[LL+99] Mong Li Lee, Hongjun Lu, Tok Wang Ling and Yee Teng Ko. "Cleansing data for mining and warehousing", In Proceedings of the 10th International Conference on Database and Expert Systems Applications (DEXA99),pages 751-760,August 1999.

[RSC99] K. Rajamani, S. Y. Sung and A Cox, "Extending the Applicability of Association Rules", In Pacific Asia Knowledge Discovery and Data Mining (PAKDD'99), Beijing, April, 1999, pp.64-73.

[SLL96] S. Y. Sung, H. Lu and Y.Lu, "On Generating Synthetic Database for Classification", 1996 IEEE International Conference on Systems, Man, and Cybernetics, Oct. 14-17, Beijing, China.

[SS95] S. Y. Sung and V. Shaw, "Mining Association Rules using Minimal Covers", Proc. of the 1995 Inter. Joint Conf. on Information Sciences, NC, USA, September 28-30, 1995.

[SWC96] S. Y. Sung, K Wang and B L Chua, "Data mining in a large database environment". In proceedings of 1996 IEEE International Conference on Systems, Man and Cybernetics (SMC'96), Beijing, China, pp. 988-993.Oct. 1996.