Bachelor's thesis

Merging Algorithm for Unified Communicator

Supervisor : Prof. Moon, Byung-Ro

Seoul National University
College of Engineering
Department of Computer Science and Engineering
Joonseok Lee

June 24, 2009

Table of Contents

Abstraction

In this thesis, I suggest an algorithm which unifies various kinds of address books in web-mails and SNS (Social Network Service), as well as in the phone. When merging two address books, there may be the same people in both. In this case, a precise algorithm to detect and merge them is essential. For this algorithm, I classified elements of address books into 6 categories: name, E-mail, phone numbers, addresses, messengers, and etc. The algorithm detects same person by summing points given by similarity between the data in both address books. I used a customized L-Distance algorithm in order to check similarity of strings, as well as equality of them. The criteria for judging similarity is set by experiments. Based on these, a precise algorithm detecting same person in two different kinds of address books, despite some difference in address book elements, is discussed.

1. Introduction

(1) Introduction to the Unified Communicator

The Unified Communicator is an application providing uniform environment for users to manage various kinds of address books on many kinds of electronic devices. Usually, many people use three to four E-mails. They also have an address book in mobile phone, and may have friends list on SNS (Social Network Services) like Facebook. Even though all of these lists share common information with a little deviation, it is truly waste of resources to manage these information separately.

However, development of unified address book was stalled due to several obstacles, despite its high demand. One of them is the fact that address books contain personal information, belonging to each company. Therefore, it was impossible to use data in various kinds of address book. Especially in Korea, unifying address book was extremely difficult due to related regulations. The other major obstacle was technical one. Specifically, address books are meaningless if they are made just by merging without recognizing same persons in each address book.

This thesis suggests a basis for solving these problems. First, I work with world-wide web-mails or SNS services which provide open source API. This removes the first problem of intellectual property rights and protection law related to personal information. This thesis focuses on developing merging algorithm with these services. The primary objective of this algorithm is finding out same persons in given two different kinds of address books. This objective is aimed to solve the second, technical problem.

Even though accuracy should be the first criteria for developing the algorithm, its efficiency is also important because this algorithm will run on embedded systems like mobile phones or PDA, which provide restricted hardware resources.

(2) Development and Test Environment

1) Hardware Environment: LG INCITE (Phone Arena)

Network: GSM type (Quad-band phone: 850/900/1800/1900 MHz)

Size: 107 x 56 x 14 mm

Bettery: Li - Polymer, 1300 mAh (equivalent to phone calls of 8 hours continuously)

Display: 240 * 400 pixel, 65536-Color, Touch-screen

Multimidia: MPEG4, 3GP, WMV, Windows Media Player

RAM: 128 MB RAM / 256 MB ROM

2) Software Requirement

Operating System: Microsoft Windows Mobile 6.1 Professional

Internet Browser: Internet Explorer Mobile

Test Environment: Microsoft Visual Studio 2008 Smart Device Emulator

(3) Previous Researches in Unified Communicators

There were many attempts to merge mobile-based address books and computer-based ones. Mitel suggested a solution to address book on computers to that on mobile smart devices. Google developed a mobile-based application allowing access to both Facebook and Blackberry on mobile devices. Nokia also distributed its own address book application.

As the above examples show, many companies have attempted to provide unified environment for managing friends list data, and to merge information on web-based address books and SNS friends list. These attempts have obtained some results in abroad. In Korea, SK Telecom, the first mobile service provider, services connection between its own SNS (cyworld.com) and mobile phones. Even though this service is available only on latest phones now, it will be expanded in near future.

However, there are some limitations in previous services. First, merging address books is restricted only on their own one or two services. Second, most of those merging services provide only simple merging solution. In most case, there are many same persons in various address books. Those services are useless in many cases, because they do not provide effective merging algorithm.

2. Background

(1) String Matching Algorithm

For the core part of merging algorithm, some technique to judge similarity of two strings as well as equivalency of them. For example, it is not difficult to find out two same persons when they share exactly same E-mail. In some cases, however, E-mails can be similar, but slightly different, although their owner is one person. For instance, a man who uses ID "clifford" in hotmail may use "clifford1" in Yahoo, due to some restrictions that any ID should contain both letters and numbers in it. These two persons can be decided as one person if they share some similarity in other information such as names or phone numbers, as well. Many researches have suggested a lot of methodologies to judge similarity of strings as below.

1) Two-way string-matching Algorithm[1)]

This paper deals with dual-way string matching. Generally, similarity is checked by moving one string as a pattern across the other one, and comparing them each time. In this paper, however, two strings are duplicated forward and backward first, and then compared each other. This paper proves this is more efficient than previous simple comparison with pattern-and-string approach. However, we need to measure how much two strings are similar. Even though this algorithm is efficient, it does not fit to our application because it does not provide similarity measurement. Thus, we dropped this for our system.

2) K Mismatch string matching Algorithm[2)]

This is a typical advanced string matching algorithm. This algorithm can find out similar strings even though string and pattern do not match exactly, but there is k mismatches. Also, Amihood Amir suggests an advanced algorithm improving previous mismatch algorithm. This algorithm fits to the Unified Communicator, in that, it provides similarity check in spite of some range of differences. The time complexity is $O(kn)$, where n is the length of longer string. In the Unified Communicator, its time complexity

1) Maxime Crochemore, Dominique Perrin, "Two-way string-matching", 1991, Journal of the ACM (JACM), Volume38 Issue 3. Publisher: ACM

2) AmihoodAmir, Moshe Lewenstein, Ely Porat, "Faster algorithms for string matching with k mismatches", 2000, Societyfor Industrial and Applied Mathematics

will be $O(n^2)$, because k value is proportional to n. It can be even seen as $O(n)$ of time complexity, because we can see k as a constant considering the fact that user ID is a short string with length of 4 to 16 and that k is smaller number than the length of ID.

This algorithm, however, is inefficient because we have to know k in advance. In this thesis, we need not similarity of two strings in the range of given k, but how much different two given strings are. In other words, we have to find k rather than k is given. Therefore, we have to test similarity of two strings for many k values in order to use this algorithm directly. This will result in increase in time complexity to $O(n^3)$, by adding one more loop. For this reason, I discarded this algorithm due to the importance of time and space complexity, especially in mobile-based environment.

3) Levenshtein distance Algorithm[3)]

This algorithm determines how much two arbitrary strings are different, by using two-dimensional array. Specifically, the L-Distance increases by 1, when one letter is different, and when there is one more or less letter in one side. For example, "abc" has a 1 of L-Distance with ”abd", "abcd", and "ab." Implementation of this algorithm is shown below:

3) Wikipedia (http://en.wikipedia.org/wiki/Levenshtein_distance)

```
int LevenshteinDistance (char s[1..m], char t[1..n])
begin
    // d is a table with m+1 rows and n+1 columns
    declare int d[0..m, 0..n]

    for i from 0 to m
        d[i, 0] := i
    for j from 0 to n
        d[0, j] := j

    for j from 1 to n
    begin
        for i from 1 to m
        begin
            if s[i] = t[j] then
                cost := 0
            else
                cost := 1

                d[i, j] := minimum(
                    d[i-1, j] + 1,       // insertion
                    d[i, j-1] + 1,       // deletion
                    d[i-1, j-1] + cost   // substitution
                )
        end
    end

    return d[m, n]
end
```

This algorithm is efficient in the respect of both time and space complexity. Both are $O(n^2)$, because it uses two-dimensional array proportional to the length of two strings. Its implementation is also simple, so this algorithm fits best to this mobile-phone based application.

(2) Address book elements for each site

In this section, items in several address books are investigated and classified. These items are essential for determining data structure of the merging algorithm. Pocket Outlook, Hotmail, Yahoo Mail, Facebook, and Flickr are investigated, and we classify common items on those services into five groups: names, E-mails, phone numbers, addresses, and messengers.

1) Names

Service Name	Address book Elements
Pocket Outlook	First name, Last name
Hotmail	First name, Last name
Yahoo Mail	First name, Last name, Nick name
Facebook	First name, Last name
Flickr	First name, Last name, Nick name

Table 1. Address book Elements related to names

As Table 1 shows, we need first name, last name, and nick name.

2) E-mails

Service Name	Address book Elements
Pocket Outlook	E-mail
Hotmail	Personal e-mail, Company e-mail, Other e-mail
Yahoo Mail	E-mail, other mail 1, other mail 2
Facebook	E-mail
Flickr	Flickr URL (E-mail ID)

Table 2. Address book Elements related to E-mails

Most address books allow saving several E-mails. In my algorithm, up to three E-mails can be stored visually, and ten more E-mails may be saved internally, when some other E-mails are found during merge process.

3) Phone Numbers

Service Name	Address book Elements
Pocket Outlook	Home, Workplace, Cell phone, Fax
Hotmail	Home, Workplace, Cell phone, Fax
Yahoo Mail	Home, Workplace, Cell phone, Fax, Other
Facebook	Other
Flickr	None

Table 3. Address book Elements related to phone numbers

Phone numbers have various elements like E-mails. Home phone, company phone, mobile phone, and fax are common, as Table 3 implies. We also store ten more other phone numbers when they are found in the process of merging.

4) Addresses

Service Name	Address book Elements
Pocket Outlook	Home, Workplace
Hotmail	Home, Workplace
Yahoo Mail	Home, Workplace
Facebook	Current Address
Flickr	None

Table 4. Address book Elements related to addresses

For addresses, home address and company address are stored generally. In some services, they are divided into sub-location according to the administrative district. That is, state, city, and street address are stored separately. In this thesis, we make use of this information for checking similarity.

5) Messenger IDs

Service Name	Address book Elements
Pocket Outlook	One
Hotmail	Windows LIVE
Yahoo Mail	Windows LIVE, Yahoo Messenger, Nateon, Buddybuddy
Facebook	Windows LIVE, Yahoo Messenger, Google Talk
Flickr	None

Table 5. Address book Elements related to messengers

Many address books provide spaces for messenger IDs. In this thesis, we use Windows LIVE, Yahoo Messenger, and Google Talk.

6) Others

Service Name	Address book Elements
Pocket Outlook	Company name, Department, Position
Hotmail	Birthday, Website, Memo
Yahoo Mail	Birthday, Website (Home/Company), Memo
Facebook	Birthday, Gender, Religion, Marriage status, Website, Schools
Flickr	Gender, Marriage status, Time zone, Memo

Table 6. Other Address book Elements

Besides the above five criteria, we may use birthday, website, and gender for checking similarity or equivalency. In addition, memo may be included for additional information for users.

3. Algorithm Suggestion

(1) Data Flow

Overall data flow is shown in Figure 1. Merge Process merges existing address book with input data, and then outputs the result as a Pocket Outlook Object Model.

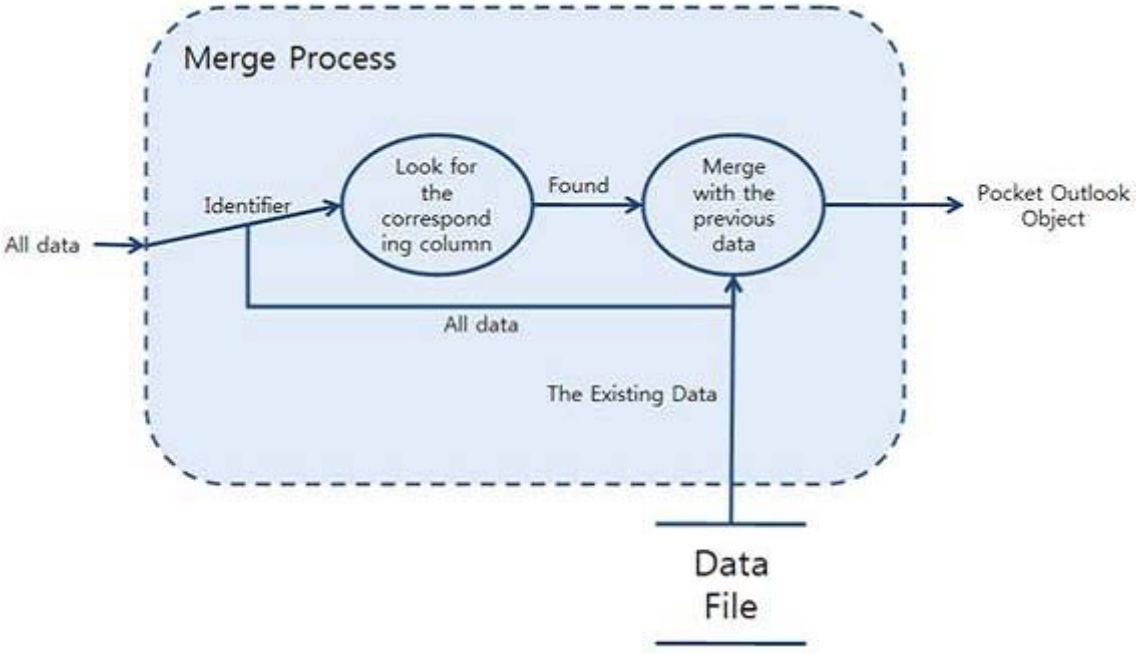

Figure 1. Level 3 Date Flow Diagram for Merge process

Data shown in Figure 1 are defined as follows:

* **All data (Input)**: Every element stored in "Data File." Contains E-mail, name, phone number, company name, for instance.
* **Identifier (Input)**: An identifier for each person in address book. In this system, E-mail address is used for this.
* **Pocket Outlook Object (Output)**: A package of elements in "All data" into the form of Pocket Outlook. In this system, data are stored with POO Model internally.

This function contains roughly two parts. First part is a process for detecting same persons, making use of identifiers among input data. (Other data also can be referenced if necessary.) For example, two persons may be considered as one person when their

E-mails are exactly same, when their E-mail is extremely similar, and when phone numbers or blog ID are same.

Second part of this algorithm is merging process. If no one was found as the same person in the first part, new records are assigned and stored as new person. If any one was found as the same person, it should be merged with the previous data about him or her. In this case, there may be some data item conflicting each other. Handling this case may be various according to policies. For example, it is entirely possible that phone numbers stored in each record are different even though they are judged as same person with E-mail address and name. We may leave the previous phone number, or replace it with new number. Some other methods like showing both may be possible.

Merged result will be printed as Pocket Outlook Object Model. This may be used by other processes which called the merge process.

(2) Detailed Implementation of Merging Algorithm

The specification of merging algorithm is as follows:

Input: address book elements for one person who will be added. Output: entire address book after adding the person in the input.

In other words, an object containing information about new person is provided as input. Then this system determines whether he or she is already stored in the existing address book. According to the result, the person will be stored either by adding or by merging with previous one. When it seems there was a record about the new person, specific data about the person will be merged. On the other hand, the person will be stored as a new object.

The most important rule of merging algorithm is that we should not merge two persons unless two or more critical information are same. This seems to be tough, but we can understand why this rule is important when we see each item in address books. Everyone has name, but there are homonym everywhere, as well. Therefore, we should not merge two persons although they have exactly same name. This is also applied to

E-mail addresses. When both ID (part before @) and domain name (part after @) are same, we can regard them as same person easily. However, two persons with same ID but different domain name may be same person, and may be not. Widely-used words like love, game, or fantastic may be used by different persons in various web sites. For phone numbers, we also can say that two persons are same if every digit is exactly same. However, phone numbers may be changed frequently, so its owner may be different if each phone number is stored at different time. Conversely, phone numbers may be totally different for the same person due to this time gap. Considering these factors lead us to conclude that we should not judge with only one element.

However, it will be almost always safe to say they are same persons, if two or more elements are same. If both name and ID are same, they would be same person with very high probability.[4] When both name and phone numbers are same, they also can be seen as same person, because it is extremely rare that two persons with same name compete for the same phone number. When more than two elements are same, probability that they are same person will approach to 1 more closely.

In this thesis, points will be added for elements showing similarity based on the above idea. Then, we will judge whether they are same person, based on the overall point sum. The threshold for determining same person would be set higher than any point which can be added by one element. By doing this, two persons will be merged only if they share at least two categories of elements. In addition, they should get points from more than two categories, if the clue is not sufficient to say that they are same person clearly. Detailed flow of this algorithm is shown below.

1) Points of each criterion

The first stage of merging algorithm is quantization of similarity of the six categories discussed in section 2.2. For each element, special characteristics should be considered here. Some items should be exactly same to be considered from same person, while some others may be slightly different even though they came from same person. Details are discussed below.

4) There could be an exception here. If the ID was named after his name, many homonym may have same ID. In this case, we should not decide hastily, and this case is also considered in this system.

A. Names: Because some web sites do not provide first name and last name separately, we compare names after concatenating them with a space. Furthermore, spells of names may be slightly different if their native language is not English. Therefore, we would give some point when names are similar as well as they are exactly same. Point table is shown in Table 7.

Case	Point
Both first and last names are exactly same.	20
Last name is exactly same, and first name is similar.	10
Both first and last names are similar.	2

Table 7. Point table for names

B. E-mails: The ID part of E-mail address can be used as a kind of identifier. The domain name part is not from the user's choice, so we can not use this information for checking same persons. Therefore, we use only ID part (before @) for comparison. As we discussed in names, partial credits are given when ID is similar, but not exactly same. This is because same person may use similar ID in various web sites. For example, a man who uses blithe@snu.ac.kr in SNU web-site may use blithe03@naver.com in another site Naver. There is an exception, though. For an E-mail shared by several employees in companies, we cannot be sure that these came from same person. For example, info@snucse.org or admin@snu.ac.kr may be used by several operators. We should not give large points in this case. Point table is shown in Table 8.

Case	Point
There is an E-mail address matching with personal E-mail.	80
There is an E-mail address matching with company E-mail.	70
There is an E-mail address matching with any other E-mail.	70
There is an ID (part of E-mail before @) matching with personal E-mail.	70
There is an ID (part of E-mail before @) matching with company E-mail.	60
There is an ID (part of E-mail before @) matching with any other E-mail.	70
There is an ID (part of E-mail before @) similar to personal E-mail.	60
There is an ID (part of E-mail before @) similar to company E-mail.	50
There is an ID (part of E-mail before @) similar to any other E-mail.	60
When the matched ID is similar to first name or last name,	-20

Table 8. Point table for E-mails

C. Phone Numbers: Before processing phone numbers, non-number characters like hyphen(-) or parentheses are removed first. There could be some variation for same phone numbers. For example, "880-4343" in Seoul can be written as "02-880-4343," "02) 880-4343," or "+82-2-880-4030." (82 is country code of South Korea, and 02 is regional code of Seoul in Korea.) Therefore, even though there are a little difference in numbers or length, they could be from a same person. So, partial credits should be awarded again. However, there are some difference here, compared to names or IDs, because phone numbers would be totally different just by changing only one digit. In order to compare without being influenced by national code and regional code, we check the last 7 digits. (Though there are many 8-digit phone numbers these days, it is still safe to say they are same persons if last 7 digits are exactly same.) Also, if the last 4 digits are exactly same, we have pretty high probability that those two persons are same one actually. This is because it is entirely possible that one person may use more than one phone, and last 4 digits tend to be assigned same in this case. Like E-mails, we give lower points for company phone numbers, because it can be shared by many employees. Fax also has lower weight because it is generally shared by many people in companies. Point table is shown in Table 9.

Case	Point
There is a phone number exactly same with cell phone number.	80
There is a phone number exactly same with home phone number.	80
There is a phone number exactly same with company phone number.	20
There is a phone number exactly same with any other phone number.	40
Last 7 digits of some phone number are same with cell phone number.	70
Last 7 digits of some phone number are same with home phone number.	70
Last 7 digits of some phone number are same with any other number.	30
Last 4 digits of some phone number are same with cell phone number.	40
Last 4 digits of some phone number are same with home phone number.	40
Last 4 digits of some phone number are same with company phone number.	10
Last 4 digits of some phone number are same with any other number.	20
There is a phone number similar to cell phone number.	25
There is a phone number similar to home phone number.	25
There is a phone number similar to any other phone number.	12
There is a phone number exactly same with fax number.	5

Table 9. Point table for phone numbers

D. Messengers: Most messenger programs use E-mail as an identifier. So, merging process is almost same with the case of E-mail. Our previous address book may have more than one messenger addresses. All of them should be compared to find out same persons. Point table is shown in Table 10.

Case	Point
There is an ID exactly matching with Windows Live ID.	70
There is an ID exactly matching with Yahoo Messenger ID	70
There is an ID exactly matching with Google Talk ID.	70
There is an ID similar to Windows Live ID.	60
There is an ID similar to Yahoo Messenger ID.	60
There is an ID similar to Google Talk ID.	60
There is an ID exactly matching with domain name in personal website.	15
There is an ID exactly matching with directory name in personal website.	15
There is an ID exactly matching with any SNS ID.	70
There is an ID exactly similar to any SNS ID.	60
There is an ID exactly matching with nick name.	25
There is an ID exactly similar to nick name.	12

Table 10. Point table for messengers

2) Summation of points

When we sum points according to the point tables discussed in the previous subsection, it is important that we should add the largest point only. Otherwise, we may give some duplicated points. For example, exactly same two phone numbers also satisfy the condition for same last 7 digits and same last 4 digits, as well. In this case, we give 80 points, highest point given to this case, according to Table 9.

3) Minus-points

All of the above tables are for giving points when two data are same or similar. Conversely, we can give some minus-points when two data which should be the same for same person are different. For instance, gender should be same for the same person. Minus-point table is shown in Table 11.

Case	Point
Different home address	-30
Different city of the company	-10
Different company name	-15
Different gender	-100
Different birthday	-30

Table 11. Minus-point table

4) Merging Threshold

Whether two persons are merged is decided by the total points they get in the previous process. Merging decision is divided into three cases. When they get more than 81 points, the algorithm merges the two persons immediately. When they get less than 60 points, the algorithm rejects to merge them. When they get points between 61 and 80, it considers they have some probability to be a same person, so it asks to users whether it should merge them.

These thresholds are determined as follows. As mentioned earlier, we should not merge two persons only with one data item. Because no single case can give more than 80 points at once, the threshold for immediate merging is set by 80. If two persons get 80 points, they may have exactly same E-mail or phone number. They have very high probability to be a same person. However, if they do not get any other points in other criteria, they have far lower probability now. So, we should not merge them unconditionally. If they have some common things other than that, we can now judge that they are actually same person with close to 100% probability.

The second threshold 60 points are determined similarly. For example, when two persons have similar Windows Live ID, they get 60 points. If they have same name, they have additional 20 points. We can think they would be the same person, but not sure because points are just 80 now. Therefore, the algorithm asks to the user whether it should merge them. In order to reduce the frequency of asking to users, we set relatively narrow gap between unconditional merging and unconditional rejection, 20 points.

All of the process we discussed in this section is for merging one person to another one. This process repeats for every existing person in the address book.

(3) Algorithm for determining String Similarity

As mentioned in the previous section, we give different points to exactly same case and similar case. It is easy to implement whether two strings are same, using API functions provided by programming languages. However, it is difficult to determine whether two strings are "similar," because we have no strict standard to judge similarity. It depends on cases. Therefore, we have to set this criterion reasonably.

In this thesis, we use L-Distance algorithm discussed in section 2. L-Distance algorithm measures the distance between two strings. It returns 0 when two inputs are exactly same, and the length of longer string when no character matches at all. In order to convert this distance value to "similarity," we divide this distance by the length of longer string. Than the index value ranges from 0 to 1. When this index value is larger than some threshold, two strings are considered similar. Otherwise, they are treated as different.

For determining the threshold, we need to investigate general pattern of IDs. That is, we need statistics on how much people tend to make variation on their ID. In order to get this data, I investigated similarity of IDs which logged in to the Hangame server (http://www.hangame.com)[5] during a month (June 2008) with same IP address. The result is shown in Table 12.[6]

Similarity	Occurrence	Rate
85% or more	8902	9%
75 to 85%	52958	52%
65 to 75%	10573	10%
65% or under	28753	28%
Sum	101186	100%

Table 12. Similarity occurrence of Multi ID users

5) In Hangame, the most famous online game portal in Korea, allows to make at most three IDs for each person. I chose this portal for this investigation because more people tend to use many IDs in game site for playing game with various positions. It is possible that there is some difference with other sites, but we assume that there is no significant difference in making IDs site by site here.

6) There could be some errors here. We assumed that IDs connected from same IP address would belong to same person. However, more than one person may share the PC at home. In this case, different ID of different person may connect from same IP address. Internet cafe may be the same case.

As we can see in Table 12, more than 60% of users multi-IDs, and more than 75% of them tend to reuse similar ID when they make new ID. Because many web-sites have their own rules for making IDs, such as requiring both letters and numbers, or minimum length. We can infer from Table 12 that one or two characters are different out of 8 characters or that one character is different out of 4 to 7 characters. For instance, a user with ID "clifford" may use "clifford2" or "clifford3" as his multi-ID.

In this thesis, the threshold is determined as 0.75 (inclusive). In other words, two strings are considered as similar when similarity calculated by L-Distance is above 0.75, and not similar otherwise.

4. Result and Analysis

(1) Analysis of Specific cases

In this section, some frequent cases in merging algorithm will be analyzed with examples. Various cases can be derived from these representative cases. We show five representative cases, skipping similar repetitive cases.

1) For exactly same two persons: Table 13 shows a case where every critical data element including name, E-mail, and phone number are same. In this case, the algorithm gives 185 points, which far exceed the threshold 80. Therefore, they are considered as same person.

	Name	E-mail	Phone No.	Company	Birthday
Person A	Nam,Young	yoyo@hotmail.com	01112344321		
Person B	Nam,Young	yoyo@hotmail.com	01112344321		

Table 13. A case where every data is exactly same

2) When almost every critical data are matched: In this case, critical data is equivalent, but some minor data are different. That is, name, E-mail, or phone number is same, but company name or birthday may be different. Even though they may get some minus-points from different items, they will be merged with high points 180 because they share the most important identifiers. In Table 14, person A and B share name, E-mail, and phone number, while company name and birthday are different. However, birthday information can be incorrect in many cases because we do not know everyone's birthday, and company name can be changed due to change of occupation. For these reasons, they are merged with this algorithm.

	Name	E-mail	Phone No.	Company	Birthday
Person A	Lee joonseok	srcw@hotmail.com	01192780945	SNU	19840809
Person B	Lee joonseok	srcw@hotmail.com	01192780945	NHN	20000101

Table 14. A case where most critical data are same

3) When critical data are similar: Unlike the case 1 and 2, some critical information may be slightly different, in this case. Specifically, name is not same, but similar in spelling. For E-mail, additional numbers can be added at the end of ID. Phone numbers can be different, but only the last 4 digits are same. In this case, L-Distance algorithm discussed in previous sections checks its similarity and gives some points. Table 15 shows this case. Person A and B have similar name, same E-mail ID, and same 4 last digits in phone number. The algorithm gives 110 points in this case, meaning that they are merged as a same person.

	Name	E-mail	Phone No.	Company	Birthday
Person A	Kwak.Wonyoung	jojo@hotmail.com	0188370947		
Person B	Kwak.Wonyoing	jojo@gura.net	01012340947		

Table 15. A case where most critical data are similar, but not same

4) When some critical data are same, but different persons actually: Sometimes different persons may have same data in some critical element. E-mail address or phone number can be exactly same, even though its owner is not same. Therefore, they should not be merged unless some other data are also implies they are same persons. Some famous E-mail addresses or phone numbers might be same. Table 16 shows an example of this case. Person A and B are different, as their different name implies. They have common E-mail address, though, because both are the president of South Korea. The algorithm gives 80 points in this case, so the user will be asked to merge them.

	Name	E-mail	Phone No.	Company	Birthday
Person A	Lee,Myungbak	president@korea.com			
Person B	Jeon,Doohwan	president@korea.com			

Table 16. A case where some data from different persons are similar

5) For totally different two persons: Last case shows two totally different persons, although some minor similarity is shown in name or phone numbers. Table 17 shows Person A and B, who have same given name, but different family name. They have same

last 4 digits in phone number. They get 40 points overall. Because it is under 60 points, the algorithm rejects to merge them.

	Name	E-mail	Phone No.	Company	Birthday
Person A	Kim, DaeJoong		0184382733		19830603
Person B	Lee, DaeJoong		0198742733		20070101

Table 17. A case with different two persons

As shown in this section, suggested algorithm in this thesis is effective for finding out same persons.

(2) Analysis of Real Samples

In this section, the accuracy of this algorithm will be analyzed with two address books actually used by the same user. Detailed information about user account is shown below:

- hotmail Address book: 182 items
- gmail Address book: 204 items
- the number of persons existing in both address books: 161 items

Table 18 shows the result of merging the above two address books.

Case	Data Amount	Occurrence
Correct for same persons		147
Correct for different persons	Sufficient	2
	Insufficient	12
Incorrect for different persons	Sufficient	0
	Insufficient	0

Table 18. Result of Merging Test

Table 18 shows that the algorithm precisely detected 147 same persons out of total 161. It fails only 14 persons to find out. However, 12 persons among them do not have

sufficient information like name, E-mail, or phone numbers. Therefore, this inaccurate result may be caused by the data itself, not from the algorithm. In addition, the fact that 21 persons only in hotmail and 43 persons only in gmail were not merged also shows accuracy of this algorithm. Thus, malfunction case was only 2 cases, implying 99% of accuracy of this algorithm.

5. Discussion

As shown in the previous section, this algorithm runs precisely both in several standard cases and on real address books. However, it may not be able to find out same persons unless sufficient amount of information is not stored in original address book. Although this problem seems to be unavoidable, it causes many problems when merging friends list of SMS, because it provides far different set of data items. Specifically, mobile phone numbers or E-mail addresses are not required in many SNS. If these information are not correct, this algorithm may fail to find out same persons in those services. In order to solve this problem, some kind of standardization of address book items may be necessary. Critical information such as name, E-mail, or mobile phone number should be required to type correctly. This requirement may be supported by regulations.

The problem may be solved by technical attempt, as well. At first, we may vary the process of giving points. In this thesis, we give points when satisfying some fixed threshold, while no point otherwise. We may classify this into two sub cases: whether the mismatch was caused by absence of data or actual mismatch despite existence. We may give different points to each case. We also can give different threshold for each case. This will improve the accuracy of this algorithm.

Also, this algorithm judges whether the two strings are similar or not, based on L-Distance algorithm. However, it is entirely possible that we can give graded points according to how much two strings are similar. In this case, however, we should be careful for designing the threshold, because it would be easy to be incorrect by giving some points to different persons and by judging them as a same person. Therefore, the amount of points for each case should be based on some statistics gathered from each web-mails and SNS sites.

Lastly, for wide use of this algorithm in many countries including native language is not English, standardization of English spelling should be considered. In other words, we should have a method to find out same persons when two different address books have same person in different language. For example, when a person's name is in Korean in one address book, while in Japanese in another one, an English-based application would

be able to find out that they are same person, only if it has a formal method converting Korean and Japanese to English. For instance, the last name 이 (in Korean) is written in LEE or YI in Korea, while the same last name 李 (in Chinese) is written in LEE or LI in China. When converting standard is prepared, both last name will be converted to LEE without any exception. This approach may improve the accuracy of this algorithm.

6. Conclusion

In this thesis, an algorithm which finds out same persons in different kinds of address books when they are merged was suggested. We show the accuracy of this algorithm by an experiment, when sufficient amount of information is provided. Especially when name, E-mail, and phone numbers are provided correctly, this algorithm can detect same persons although they have some variation.

However, when the data is not sufficient in any address book, merging process can fail to merge data from same persons. This problem is not a technical one, but a political one. In other words, should we decide that two persons are same, when we have little, but same information about them? If we say YES, we may reduce errors of "saying different persons for actually same person," but the other errors of "saying same person for actually different persons" will increase. Because I think the latter error is more serious, I decided not to merge in such a case. For not merged same persons, it just causes some annoying task to merge them manually. For merged different persons, however, it may corrupt each item in one of them. It would be impossible to recover these corrupted information after merged.

As I declared before, this kind of research is in fundamental level. Although some researches may be conducted by some mobile-phone companies or SNS providers, the result tends to be hidden due to the patent reason. However, according to the current trend of open API and open sources, researches on effective merging algorithm for address books should be conducted in the near future.

7. Bibliography

[1] Maxime Crochemore, Dominique Perrin, “Two-way string-matching”, 1991, Journal of the ACM (JACM), Volume38 Issue 3. Publisher: ACM

[2] AmihoodAmir, Moshe Lewenstein, Ely Porat, “Faster algorithms for string matching with k mismatches”, 2000, Societyfor Industrial and Applied Mathematics

[3] Cormen, Leiserson, Rivest, and Stein. “Introduction to Algorithms, 2nd ed.”, 2001, MIT Press

[4] Gonzalo Navarro, “A Guided Tour to Approximate String Matching”, 2000, University of Chile

[5] Quinn, Bob(Author), Shute, David K. (Author), Shute, Dave (With) “Windows Sockets Network Programming”, Addison-Wesley Professional