Dipper: A Data Integration with Privacy Protection Environment

Rangsipan Marukatat

***Abstract*—We have developed Dipper, a data integration with privacy protection environment, aiming to tackle the following challenges. Firstly, records can be matched albeit typographical differences between their attribute values. Secondly, they can be matched albeit some attribute values being suppressed to protect the privacy of an individual. Dipper enables data providers to suppress sensitive attributes before exporting their data sets. It then integrates the data sets based on matching rules specified by the users. Its simple graphical user interfaces (GUIs) guide the users through a few easy steps to complete their tasks.**

***Index Terms*—data integration, privacy protection.**

I. Introduction

Many real-world applications process data from multiple data sets. For example, a marketing department devises new marketing campaigns by analyzing customer profiles (gender, age, etc.) and transaction data. To consolidate multiple data sets, one needs to identify records that *match* each other, i.e. they refer to the same entities. These records can be merged or duplicate ones can be eliminated. This process has several names such as data integration, data linkage, record linkage, and deduplication. In data mining and data warehousing, it crucially helps increase the dimensionality of data.

If every record contains *hard* key attributes such as social security number, barcode, or ISBN, integrating the data sets is similar to joining database tables. It is a trivial task since the attributes' values are often error-free; otherwise, a transaction would have failed and the data would not have been stored in a data repository. However, disparate data sets do not always use the same hard keys. Instead, the integration often relies on *composite* keys such as a composite of first name, surname, and date of birth attributes. These attributes' values are more likely to be misspelled or written in different styles. Finding algorithms that are robust to typographical differences is a challenge in data integration research.

Another challenge in data integration stems from the need to protect privacy of an individual, particularly when data providers and data collectors are different parties. The data providers need to remove or transform personal identifiable attributes before giving out the data. At the same time, the lack or the anonymity of such attributes makes the integration task even more complicated.

Manuscript received June 27, 2008.

R. Marukatat is with the Department of Computer Engineering, Faculty of Engineering, Mahidol University, THAILAND. (phone: +662-889-2138 ext 6251-2; fax: +662-889-2138 ext 6259; e-mail: egrmr@mahidol.ac.th).

We have developed a data integration tool called "Dipper" (**D**ata **i**ntegration with **p**rivacy **p**rotection **e**nvi**r**onment). It consists of *Data Exporter* and *Data Integrator* modules. The former suppresses sensitive attributes and exports a data set. The latter integrates data sets according to user's specified rules. The rest of this paper is organized as follows. Section II reviews related techniques and tools. Section III describes Dipper framework. Section IV presents data integration tasks performed by Dipper. Section V finally concludes the paper.

II. Data Integration Techniques and Tools

Record matching requires comparison in two levels [4], [5]. At low level, each pair of attributes (one from each record) is compared. Since attributes' values may contain typographical errors, string similarity test is typically applied instead of the exact equality test. At high level, similarity between records is determined by combining the individual attribute similarity scores. For two data sets having n and m records, nm record matchings are performed. This poses scalability problem in real practice. TAILOR [5] and Febrl [1] handle large data volume by using some key attributes to (1) split the records into mutually exclusive blocks; or (2) sort the records and move a sliding window over them. As a result, the matching space is shrinked to one block/window at a time. [1] found that comparing and merging the records are scalable tasks which can be executed efficiently by parallel processors.

With the increasing concern about privacy, data integration tools have incorporated privacy protection and blindfolded data integration mechanisms. The first mechanism suppresses sensitive, identifiable attributes such as patient name or social security number. The second mechanism matches records by comparing the suppressed attributes. [9] suggested one-way hash function as a simple suppression method, but a drawback is that any marginal discrepancy between the original attribute values will be enormously magnified.

An alternative method, minimal knowledge n-gram similarity test, was proposed by [2]. It first generates a power set of encrypted n-grams for each attribute value. When the attributes are compared, their power sets are intersected and the attribute similarity score is calculated from the common n-grams. [8] proposed finding a cluster of strings similar to each attribute value from a public reference table. The strings are encrypted. Attribute similarity is determined in the same way as the n-gram similarity test. However, these methods are memory inefficient because they replace a single attribute value with a set of encrypted data.

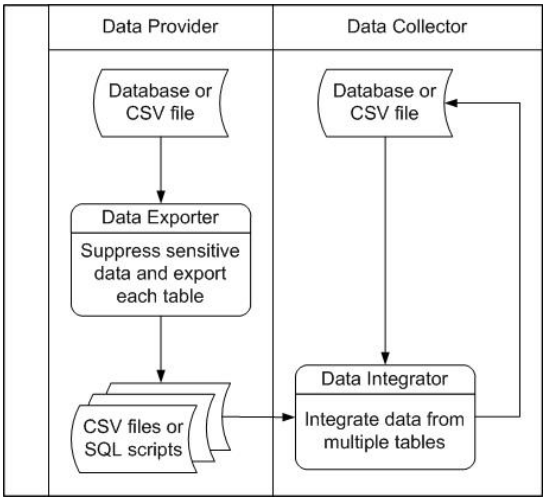

Fig. 1. Data export and data integration tasks performed by data provider and data collector, respectively

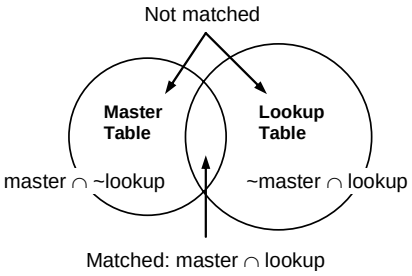

Fig. 2. Possible outcomes of record matching

III. Dipper Framework

Our **D**ata **i**ntegration with **p**rivacy **p**rotection **e**nvi**r**onment (Dipper) comprises two modules, *Data Exporter* and *Data Integrator*. These modules can be used by data providers and data collectors as illustrated by Fig. 1.

Dipper's GUI was designed to be as simplest as possible. We observed that average users are most comfortable to work with Microsoft Excel-style spread sheets. So, Dipper presents data in tables. Users can edit table cells or transfer data within and between the tables via cut-and-paste operations. The main Data Exporter and Data Integrator windows are divided into subject panels, ordering from left to right and top to bottom. Such interface is self-explanatory; it easily guides the users through to complete their tasks.

A. Data Exporter

The main Data Exporter window is shown in Fig. 3. To export a data set, the user loads data from a comma-separated-value (CSV) file or a database table. Then he/she might suppress sensitive attributes. The following suppression methods were implemented. They are based on cryptographic hashing:

- *MD5* (RFC 1321). MD5 takes a message of arbitrary length as input and produces a 128-bit message digest as output.
- *SHA-1* (FIPS 180-2). SHA-1 takes a message with a maximum length of 2^{64} bits and produces a 160-bit message digest as output.

Finally, the data set can be exported as a CSV file or SQL script consisting of `CREATE TABLE` and `INSERT INTO TABLE` commands.

B. Data Integrator

The main Data Integrator window is shown in Fig. 4. To integrate two data sets, the user loads data into Master and Lookup tables. Then he/she needs to define matching rules (see Fig. 5) and choose mode of integration. The results are presented in Master table (it is assigned a new table name, so as to not replace the old table). Record similarity scores are reported. The user may correct the result manually, by editing the table. It is either exported as a CSV file or SQL script, or saved to the database.

Matching records in Master and Lookup tables yields three possible outcomes, as illustrated by Fig. 2. If the tables are *completely joined*, the result will include {master ∩ lookup}, {master ∩ ~lookup}, and {~master ∩ lookup}. In some cases, the user focuses on entities in the Master table and wants to look up additional information about these entities in the Lookup table. Data integration can be executed in *lookup* mode, producing {master ∩ lookup} and {master ∩ ~lookup} as the result. Thanks to fewer record matchings and thus lower computational overhead, the lookup mode is set as default one.

Dipper attempts to match records *A* and *B* against a number of matching rules specified by the users. A matching rule may have several clauses, connected by logical AND operators. It can be written as follows:

comparator(A's attribute, B's attribute) $\geq s_1$
AND
comparator(A's attribute, B's attribute) $\geq s_2$
AND
…
AND
comparator(A's attribute, B's attribute) $\geq s_k$.

A comparator treats attribute values as strings. It calculates string similarity score which falls within a [0, 1] interval. If the score is not below a threshold *s*, then the attributes are considered similar. Once every clause in rule is true, *B* and its average matching score will be memorized as a candidate match for *A* and the next record will be tested. The best match for *A* is then the one with the highest average matching score.

The following string comparators are used by Dipper:

- *Levenstein*. Levenstein measures a distance between two strings as the minimum edit operations required to convert one string to the other. An edit operation is the insertion, deletion, or substitution of a character. The similarity score is then calculated as a reverse measure of the distance.
- *Jaro* and *Jaro-Winkler* [10]. Jaro similarity score is a function of common characters between both strings (the same characters appearing at nearby positions) and the transpositions from one string to the other. Its variant, Jaro-Winkler, adds weight to account for the length of the common prefix, because typographical errors are more likely at the tail-end of the strings and when the strings are longer.

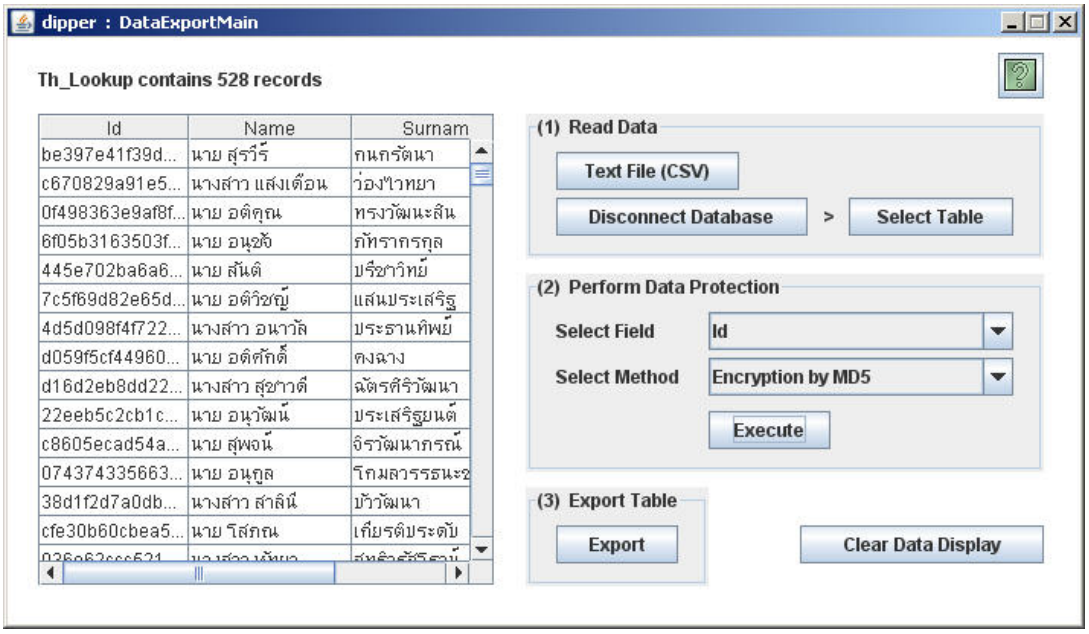

Fig. 3. Main Data Exporter window ("Id" has been encrypted by MD5)

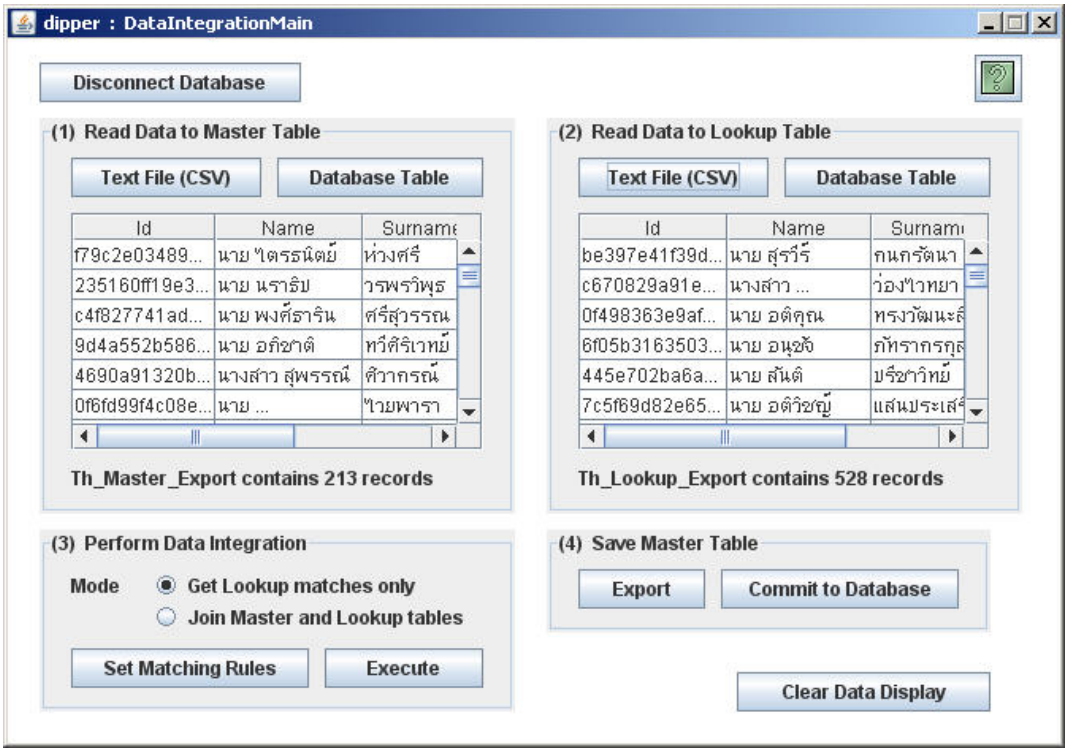

Fig. 4. Main Data Integrator window

– *Monge-Elkan* [6]. It is a variant of Levenstein comparator which allows affine costs for edit operations. For example, successive insertions yield lower costs than separate insertions.

It is possible that a string consists of tokens, or sequences of characters delimited by punctuations. The arrangement of tokens in any two strings may also differ, for example, "Mary Jane Watson" and "Watson, Mary J.". To tackle this, one should compare every token in one string with every token in the other string. This solution was implemented by Cohen *et al* as Level-2 comparators [3], [11]. They are also employed by Dipper.

C. *Dipper's Implementation*

Dipper is based on opensource technology. It was written in Java. Its execution requires Java Runtime Environment (JRE) version 1.5 or higher. Cohen's SecondString package [11] are employed for string comparison. Dipper was tested with MySQL, but users can work around the other types of databases by exporting tables and processing flat CSV files.

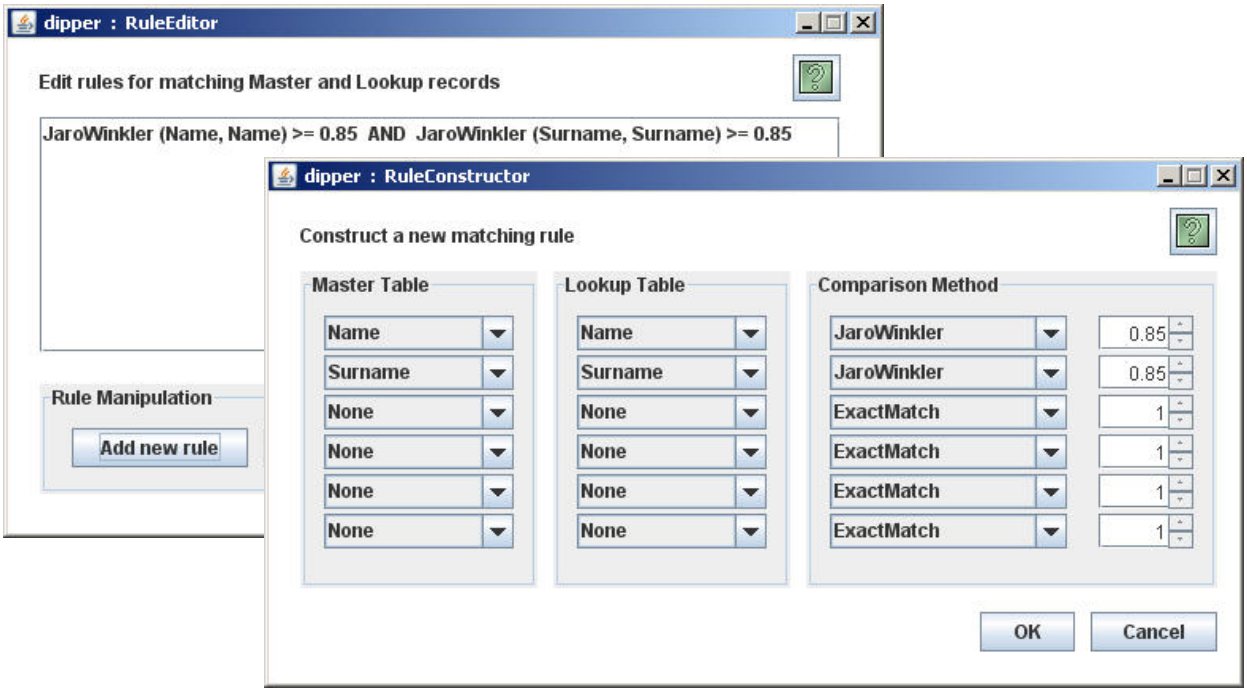

Fig. 5. Rule Editor and Constructor windows

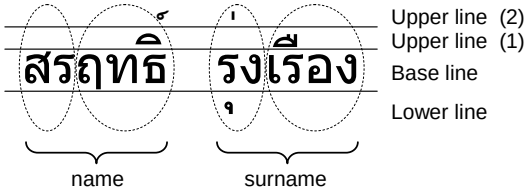

Fig. 6. Thai name and surname, each having 2 syllables (of length 2 and 5 for name's, 4 and 5 for surname's)

IV. Experiments

Dipper lets users construct and refine matching rules by themselves. Hence, the following experiments serve only as general guidelines about how each comparator performs. The performance based on English data had already been reported by the SecondString developers themselves [3]. In their study, a number of real and synthetic data were tested. Among the four comparators used by Dipper, they found that Monge-Elkan performed better than the others on average. But there were a few cases where it lost out to Jaro and Jaro-Winkler.

We conducted further experiments by using Thai data. Two data sets, Th_Lookup and Th_Master, were retrieved from the Faculty of Engineering at Mahidol University, Thailand. Each data set was originally clean and contained no duplication. There were 528 student records (ID, name, surname, and major) in Th_Lookup, and 213 senior project records (ID, name, surname, project, and advisor) in Th_Master. About 50% of the records in Th_Master found their matches in Th_Lookup.

A. Experimental Setup

Adapting the methodology used by [1], we added errors to name and surname attributes of some records as follows.

- Add a character randomly.
- Delete a character randomly.
- Replace a character with its neighboring character on the keyboard.
- Transpose two characters randomly.
- Merge name and surname.
- Merge name and surname, then swap the values.

The first four data transformations were applied to evaluate Levenstein (L), Jaro (J), Jaro-Winkler (JW), and Monge-Elkan (ME) comparators. The other two data transformations were applied later to evaluate the Level-2 versions of these comparators. They are L2-L, L2-J, L2-JW, and L2-ME, respectively.

In each experiment, we constructed a single matching rule which compared name and surname attributes, by using the same comparator and similarity threshold. For example,

JaroWinkler(name, name) >= 0.85 AND
JaroWinkler(surname, surname) >= 0.85 .

B. Results

Thai alphabet is far more complex than English one. It includes 44 consonants, 18 vowel symbols which make up 32 vowels, 4 tone marks, and 2 diacritics [7]. Thai text is written in four lines, as illustrated by Fig. 6. There are certain rules for mixing consonants, vowel symbols, tone marks, and diacritics together. The length of a Thai syllable can vary from 2 to 10 characters. A name is usually up to 5 syllable long; a surname might be a little bit longer.

Fig. 7 shows the accuracy, false positive (FP), and false negative (FN) rates of each comparator against 0.75, 0.85, and 0.95 thresholds. We observed that comparators that takes patterns of characters into account, such as Jaro and Jaro-Winkler, performed better than comparators that processes characters one by one, such as Levenstein and Monge-Elkan.

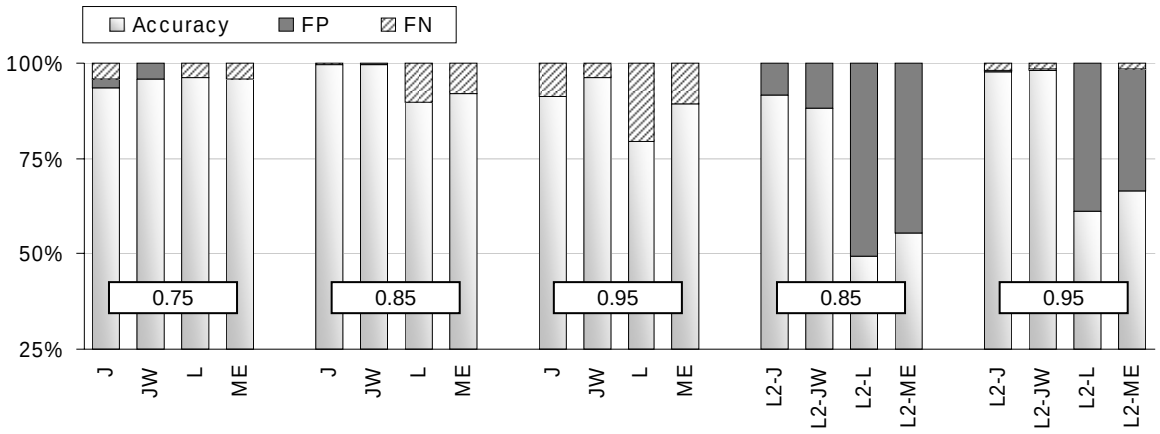

Fig. 7. Performance of each comparator at 0.75, 0.85, and 0.95 similarity thresholds

When name and surname were compared separately, Jaro and Jaro-Winkler performed slightly better than the other two comparators. At the 0.85 threshold, they matched the records nearly 100% accurately, while Levenstein and Monge-Elkan failed to match some records (about 10% false negative rate). At the 0.95 threshold, the false negative rates increased for all the comparators.

When name and surname were merged into one attribute, Level-2 Levenstein and Monge-Elkan were very optimistic and incurred very high false positive rates. This is because edit operations at the boundaries of consecutive syllables in one string could result in new syllables that exist in the other strings. Therefore, strings that consist of multiple substrings are likely considered similar by these comparators.

V. Conclusion

We have presented Dipper, a Data integration with privacy protection environment, which comprises tools for exporting and integrating data. To protect individuals' privacy, users can suppress sensitive attributes prior to the exporting. Dipper merges records from two data sets according to user-specified matching rules. It supports the well-known Levenstein, Jaro, Jaro-Winkler, and Monge-Elkan string comparison methods. We tested these methods with Thai data and found that, given appropriate similarity thresholds, Jaro and Jaro-Winkler were able to match most of the records correctly.

A limitation of Dipper is that its suppression methods are based on cryptographic hashing. If data integration requires the comparison of suppressed attributes, then a pair of records will match only if their original attribute values are exactly equal. This does not leave room for even marginal errors.

Besides its functionality, Dipper's worth is its simple user interface. Bearing in mind that average users are most familiar with spread sheet presentation and cut-and-paste operations, Dipper imitates such working environment. Morever, due to self-explanatory interface, users can easily go through a few steps to complete their tasks.

We aim to improve Dipper in two directions. The first one is adding suppression methods that do not cryptographically transform the data. An alternative solution, as discussed in Section II, is replacing old values with new ones associated in some ways with common reference data. But for this solution to be practical, the storage space of the new values must not overly exceeds that of the old ones. The other improvement is adding attribute comparison methods that are more advanced and, perhaps, specific to Thai language.

Acknowledgment

Dipper's foundation classes were implemented and tested by Angkana Chanjirakitti, Parunya Sonthipong, Siwapong Srisodsai, Wanna Kanuwattana, and Yosita Thitiangkool.

References

[1] P. Christen, T. Churches, and M. Hegland, "Febrl – A parallel opensource data linkage system," in *Proceedings of the 8th Asia-Pacific Conference on Knowledge Discovery and Data Mining (PAKDD)*, Sydney, Australia, 2004.

[2] T. Churches and P. Christen, "Some methods for blindfolded record linkage," *BMC Medical Informatics and Decision Making*, 4(9), 2004.

[3] W. W. Cohen, P. Ravikumar, and S. E. Feinberg, "A comparison of string distance metrics for name matching tasks," in *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI)*, Acapulco, Mexico, 2003.

[4] M. G. Elfeky, V. S. Verykios, and A. K. Elmagarmid, "TAILOR: A record linkage toolbox," in *Proceedings of the 18th International Conference on Data Engineering (ICDE)*, San Jose, CA, 2002.

[5] L. Jin, C. Li, and S. Mehrotra, "Efficient record linkage in large data sets," in *Proceedings of the 8th International Conference on Database Systems for Advanced Applications (DASFAA)*, Kyoto, Japan, 2003.

[6] A. E. Monge and C. P. Elkan, "The field matching problem: Algorithms and applications," in Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD), Portland, OR, 1996.

[7] T. Karoonboonyanan, "Standardization and implementations of Thai language," National Electronics and Computer Technology Center (NECTEC), Thailand. http://www.nectec.or.th/it-standards/thaistd.pdf

[8] C. Pang and D. Hansen, "Improved record linkage for encrypted identifying data," in *Proceedings of the 14th Annual Health Informatic Conference (HIC)*, Sydney, Australia, 2006.

[9] C. Quantin, *et al*., "How to ensure data security of an epidemiological follow-up: Quality assessment of an anonymous record linkage procedure," *International Journal of Medical Informatics*, 49(1), pp. 117-122, 1998.

[10] W. E. Winkler, "Advanced method for record linkage," in *Proceedings of the Section of Survey Research Methods, American Statistical Association*, pp. 467-472, 1994.

[11] Second String Project Page, http://secondstring.sourceforge.net/.