2011 International Conference on Information Communication and Management
IPCSIT vol.16 (2011) © (2011) IACSIT Press, Singapore

Matching Entities from Bilingual (Thai/English) Data Sources

Rangsipan Marukatat [+]

Department of Computer Engineering, Faculty of Engineering, Mahidol University

Abstract. We develop a framework to match records from multiple data sources that belong to the same entities. In real practice, some data sources may store the data in Thai but some may store the data in English. Therefore, Thai-version keys are romanized. Then, English-version keys are compared by using approximate string comparators and rule-based decision function. Experimental results are reported. Problems, challenges, and possible research directions are also presented.

Keywords: entity matching, record matching, entity names

1. Introduction

Many information systems these days process data from many data sources. Therefore, one essential task is identifying records from disparate sources that refer to the same entities. This task has several names such as entity matching, record matching, record linkage, or data deduplication. After matched records are found, they are merged to increase the data's dimensionality, or duplicated ones are removed. In Thai, an entity's name may be spelled with slight variations such as "พิสณุ", "พิสนุ", "พิศณุ", "พิศนุ", or "พิษณุ. This may be due to misunderstanding or typing errors. Some data sources store the names in Thai characters but some store them in English characters. Although an official romanization standard has been set [1], it is not widely and strictly followed in real practice. Hence, "พิสณุ" (or its variant) may be written in English as "Pitsanu", "Pissanu", or "Pisanu". Given such scenarios, our record matching task must be able to compare an entity's name written in any different variations, either in Thai or English.

We have been developing a data integration tool for Thai/English naming context. In Section 2, we offer brief information about Thai characters, the romanization from Thai to English, and a few challenging issues. In Section 3, we review record matching methods implemented in our tool. In Section 4, experimental results are presented and discussed. Finally, Section 5 concludes the paper.

2. Thai Characters and Romanization

Shown in Fig. 1, there are 44 Thai consonants in 21 phonetic groups, 18 vowel symbols that make up 32 vowels, 4 tone marks, and 2 diacritics (sound shortener and sound killer) [2]. Thai characters are printed in four lines. Forward characters are printed on the base line, occupying horizontal space. Dead characters are printed above or below the forward characters. According to Thai encoding standard, TIS620, characters are read from left to right. If there are multiple characters in one column, they are read in the following order: forward character; dead character on the lower line; dead character on the upper line; and dead character on the top line.

A Thai syllable typically consists of a leading consonant, a vowel, a final consonant, and a tone. There is no space between syllables and determining the correct boundary of each one is a non-trivial task. Indeed, it is one of challenging issues in linguistic and information retrieval research (e.g. [3], [4]).

[+] Corresponding author. Tel.: +662 889 2138 ext 6251; fax: +662 889 2138 ext 6259.
E-mail address: egrmr@mahidol.ac.th.

Thai proper names are usually derived from Indian origin languages, either Pali or Sanskrit. Through a series of conversions from the original language, a name can be written in a few subtly different ways. Thus, typographical errors or inconsistencies are very common. To write a name in Roman or English characters, Thai romanization is performed. According to the government proclamation, as reported in [5], romanization rules are based on sound transcription. Sounds are mapped to certain English characters that give the closest pronunciation, disregarding tone mark and diacritic. Therefore, "**รุ่ง**" and "**รุ้ง**", pronounced in different tones, are converted into "Rung". The following names "**พงก์**", "**พงศ์**", and "**ภงค์**" are all converted into "Phong" as "**พ**" and "**ภ**" are in the same phonetic group while "**ก**", "**ศ**", and "**ค**" are silenced by sound killers. A few attempts at automatic Thai romanization are [4], [6], and [7], among others.

Sometimes Pali/Sanskrit romanization is applied in order to retain the meaning in Pali/Sanskrit, despite inexact sound mapping. For example, the official name of Thailand's international airport "**สุวรรณภูมิ**" is "Suvarnabhumi" by Sanskrit romanization, rather than "Suwannaphum" by Thai romanization.

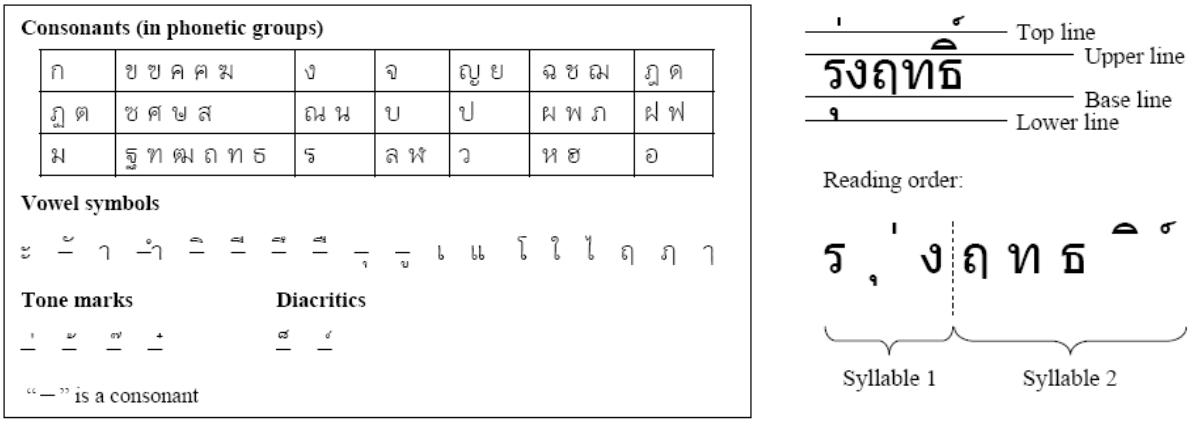

Fig. 1: Thai characters and their reading order

3. Approximate Record Matching

Suppose that there are two data sources A and B. The purpose of record matching is to identify record(s) b in B that belong to the same entity as record a in A. The set of matched records can be written as $A \times B$ or $\{(a, b) \mid \forall a \in A, \forall b \in B\}$. To determine whether a and b belong to the same entity, a's keys $\{K_{a1}, K_{a2}, ..., K_{an}\}$ are compared with the corresponding b's keys $\{K_{b1}, K_{b2}, ..., K_{bn}\}$.

In deterministic matching, equality test is performed for each pair of K_a and K_b. This is practical for hard keys such as ISBN, barcode, or social security number. Assuming that A and B are taken from existing data repository, these key values are likely error-free. Otherwise, the data/transactions would not have been stored in the first place.

However, different data sources may use different hard keys. Some may not use hard key at all. So the matching requires other keys such as name and surname. Similarity test is typically employed because key values may be misspelled or written in different styles. This is called approximate matching. In approximate matching, the similarity between K_{ai} and K_{bi}, $1 \le i \le n$, is measured, yielding δ_i. A decision function takes $\{\delta_1, \delta_2, ..., \delta_n\}$ and produces a record matching score (Δ) for a and b.

3.1. String Comparators

To measure the similarity (δ) between K_a and K_b, both key values are treated as strings. The similarity score is normalized to [0,1] range. String comparators used in this research are as follows.

Levenshtein: This comparator calculates an edit distance based on the minimum number of insertions, deletions, and substitutions of characters that convert one string into another. Each operation incurs one unit cost. The similarity score is a reverse measurement of the distance.

Monge-Elkan: Like Levenshtein, this comparator calculates a similarity score based on edit distance. But it assigns decreasing costs to successive operations [8].

Jaro-Winkler: This comparator counts identical characters in corresponding and nearby positions (not farther than half of the length of the shorter string) in both strings, and the transpositions from one string to another. Weights are added according to the characters' positions because characters at the tail-end of the string are more likely to differ than those at the beginning of the strings [9].

Recursive comparator: Strings K_a and K_b may consist of tokens or substrings delimited by punctuations, arranged in different orders. For example, their values might be "Harry James Potter" and "Potter, Harry J.". The recursive method compares every token in one string with every token in the other, using any distance-based comparator (e.g. Levenshtein, Monge-Elkan, or Jaro-Winkler), and calculates the total score [10].

The above comparators compare Thai-Thai or English-English strings. In case that one of them is in Thai and the other is in English, the Thai string is converted into English by using Aroonmanakun's romanization ([4], [6]).

3.2. Decision Function

A decision function combines n comparison results $\{\delta_1, \delta_2, \ldots, \delta_n\}$ and gives a record matching score. This function can be simple linear regression, expectation-maximization (EM), decision tree, support vector machine, or user-defined rules. In this research, our decision function is rule-based. A user can specify a set of matching rules. Each rule contains clauses connected by logical AND operators. For example,

$$\begin{aligned}
&\textit{Rule 1: } \textit{Jaro-Winkler}(K_{a1}, K_{b1}) \geq s_{11} \quad \textit{AND}\\
&\qquad \textit{Jaro-Winkler}(K_{a2}, K_{b2}) \geq s_{12}\\
&\textit{Rule 2: } \textit{Levenshtein}(K_{a1}, K_{b1}) \geq s_{21} \quad \textit{AND}\\
&\qquad \textit{Levenshtein}(K_{a3}, K_{b3}) \geq s_{23} \quad \textit{AND}\\
&\qquad \textit{Jaro-Winkler}(K_{a4}, K_{b4}) \geq s_{24}
\end{aligned}$$

A threshold s is given for every clause in a rule. A matching score (Δ) according to rule R is the average of all similarity scores (δ's) in that rule. To find a record in data set B that best matches a, the following is performed:

```
(1)  Candidate_Set = ∅
(2)  For each record b in data set B {
(3)     For each rule R {
(4)        If every clause in R is true {
(5)           Calculate matching score Δ
(6)           Add b to Candidate_Set
(7)           Break (i.e. skip remaining rules)
(8)        }
(9)     }
(10) }
(11) Choose b with the highest Δ from Candidate_Set
```

If we want multiple records in B to match a, then candidates can be sorted by their matching scores and the first few can be chosen. On the other hand, if no match is found, then decision rules may be adjusted. The above comparators and matching strategy are implemented in our data integration software [11].

4. Experiments

Our Master data set contained 1,480 records of our Engineering students. Each record consisted of ID, name and surname in Thai (TH_name and TH_surname), name and surname in English (EN_name and EN_surname), and major.

In the first experiment, we generated five Lookup data sets, each containing 200 records randomed from the Master set. We matched records by using Thai-version names in Lookup and English-version names in Master. TH_name and TH_surname were romanized into RO_name and RO_surname. The same comparator,

Jaro-Winkler (JW), Levenshtein (L), or Monge-Elkan (ME) were used for comparing names and surnames, e.g. *JaroWinkler(EN_name, RO_name) ≥ 0.1 AND JaroWinkler(EN_surname, RO_surname) ≥ 0.1*. The threshold was set to 0.1 in order to allow both correct and incorrect matching.

In the second experiment, we added typographical errors to 50% of records in both Master and Lookup sets, and performed record matching as in the first experiment. Random errors were added to EN_name and EN_surname in Master, as well as to TH_name and TH_surname in Lookup. The number of induced errors ranged from 1 to 4 per record. An error can be one of the following:

- Repetition, e.g. from "Rung" to "Run**n**g"
- Substitution with neighboring character on the keyboard, e.g. from "R**u**ng" to "R**y**ng"
- Substitution with look-alike character, e.g. from "Run**g**" to "Run**q**"
- Substitution with shifted/unshifted character, e.g. from "**รุ่ง**" to "**ณุ่ง**" (shift and "**ร**" yields "**ณ**")

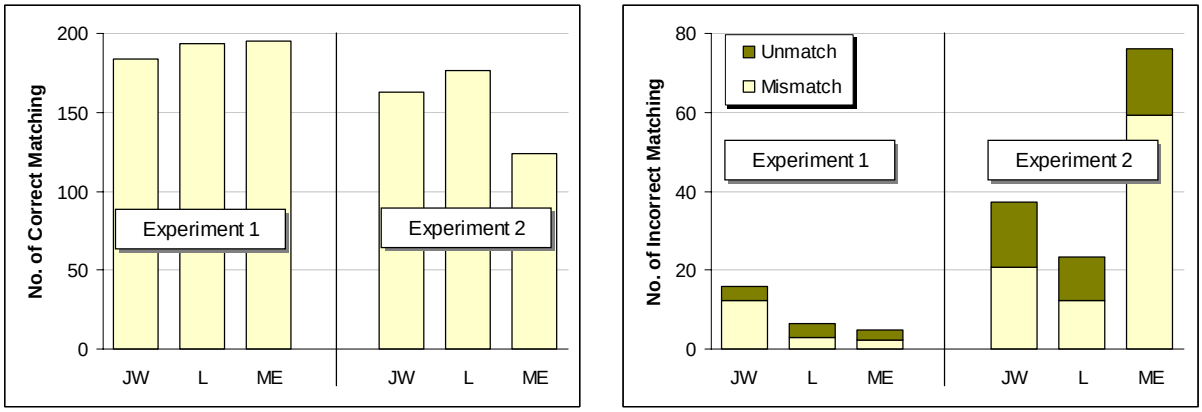

Fig. 2: Average numbers of matched, mismatched, and unmatched records in both experiments

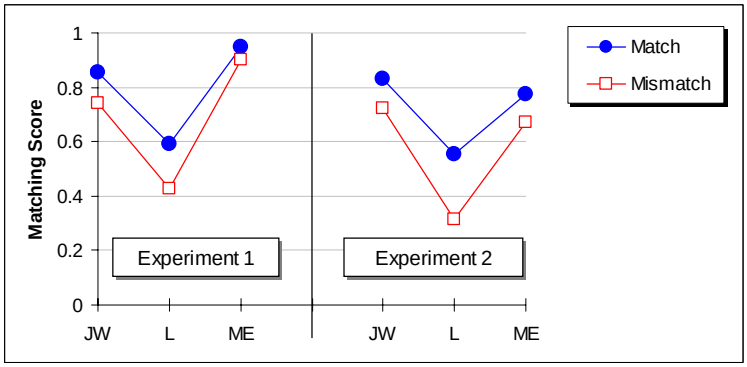

Fig. 3: Average matching scores of matched and mismatched records in both experiments

4.1. Results and Discussion

Fig. 2 displays the average numbers, across 5 Lookup data sets, of correctly matched, mismatched, and unmatched (i.e. no matching was found) records in both experiments. Fig. 3 displays the matching scores of matched and unmatched cases. In the first experiment, Levenshtein and Monge-Elkan performed better than Jaro-Winkler. Monge-Elkan was more optimistic than Levenshtein as it gave higher matching scores (Fig. 3). We found that all comparators left nearly identical sets of unmatched records. Thai-version names/surnames in these records were segmented and romanized incorrectly. As a result, whole syllables were missing from RO_name/RO_surname, making them too much different from their corresponding EN_name/EN_surname in the Master set.

As typographical errors were induced in the second experiment, the number of incorrect segmentation and unpronounceable strings increased, leading to more unmatched cases. On the other hand, some changes in Thai-version names/surnames did not affect the romanization, e.g. "**พงก์**" and "**ภงค์**" were still mapped to the same romanized string "Phong". There was a little drop in Levenshtein's and Jaro-Winkler's matching performance. But Monge-Elkan tended to be too optimistic and gave too many mismatched records.

5. Conclusion and Future Work

Our research aims to tackle approximate record matching, where entity names may be stored in different variations, either in Thai or English characters. We convert the names stored in Thai characters into English by using transcription-based romanization, and compare them with the English names by using well-known string comparators and rule-based decision function. Experimental results showed that these comparators were quite effective. But there were problems with automatic romanization, especially when typographical errors were present and syllables' boundaries could not be determined. We will investigate other approaches for string segmentation and romanization. Alternatively, the integration program should be able to recognize and cleanse some errors beforehand. Moreover, we will apply supervised learning techniques to tackle cases where Thai names are translated, e.g. from "ตึกช้าง" to "Elephant Building", rather than romanized to "Tuek Chang"; or where Thai and English names are neither romanization nor translation of each other, such as "กรุงเทพมหานคร" and "Bangkok".

6. References

[1] N. Kanchanawan. Romanization, transliteration, and transcription for the globalization of the Thai language. *The Journal of the Royal Institute of Thailand*. 2006, 31(3): 832-841.

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

[3] C. Wutiwiwatchai and A. Thangthai. Syllable-based Thai-English machine transliteration. *Proceedings of the 2010 Named Entities Workshop, ACL 2010*. Upsala, Sweden, 2010, pp. 66-70.

[4] W. Aroonmanakun and W. Rivepiboon. A unified model of Thai romanization and word segmentation. *Proceedings of the 18th Pacific Asia Conference on Language, Information and Computation (PACLIC)*. Japan, 2004, pp. 205-214.

[5] The Royal Institute of Thailand. Principles of romanization for Thai script by transcription method. *Eighth United Nations Conference on the Standardization of Geographical Names*. Berlin, Germany, 2002.

[6] W. Aroonmanakun. Thai romanization (software). http://www.arts.chula.ac.th/~ling/tts.

[7] A. Tangverapong, A. Suchato, and P. Punyabukkana. Romanization of Thai proper names based on popularity of usages. In: T. Theeramunkong, et al (eds.). *PAKDD 2009, Lecture Notes in Computer Science Vol. 5476*. Springer. 2009, pp. 580-587.

[8] A. E. Monge and C. P. Elkan. The field matching problem: algorithms and applications. In: E. Simoudis, et al (eds.). *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD)*. AAAI Press. 1996, pp. 267-270.

[9] W. E. Winkler. *The state of record linkage and current research problems*. Technical Report RR/1999/04. US Bureau of Census, Washington DC, 1999.

[10] W. W. Cohen, P. Ravikumar, and S. E. Feinberg. A comparison of string distance metrics for name matching tasks. *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. Mexico, 2003, pp. 73-78.

[11] R. Marukatat. Dipper: a data integration and privacy protection environment. *Proceedings of the International MultiConference of Engineers and Computer Scientists (IMECS)*. Hong Kong, 2009, pp. 750-754.