Transliteration Mapping in Intercultural Web Search

ABSTRACT

This paper is about searching for romanized transliterations of terms from languages written with non-Roman scripts, particularly Mandarin Chinese and Hindustani. The Roman script remains a writing system in many languages, especially Western European languages like French. In a globalizing world, the Roman script is also being used to transliterate many languages that traditionally use alternative writing systems. Search engines are able to suggest alternate spellings for words in languages that use the Roman script as a writing system, but they do not yet look for all possible transliterations of the same word in languages that use the Roman script only for transliteration purposes. This paper describes problems and solutions regarding Romanized transliteration of terms originally from Mandarin Chinese and Hindustani.

Keywords

Transliteration, Search Engine, Mandarin, Hindustani

INTRODUCTION

People from different cultures often seek information that may have to do with another culture. A businessman from Boston planning to travel to Beijing may wonder which restaurant has the best Peking Duck. A Beijinger may try to buy products from the many multinational companies that advertise in China. An Italian fan of Indian (Hindustani-language) films may be looking for a film song but may feel compelled to search in Roman script. All of these are examples of *intercultural search*.

When intercultural search takes place on the Internet, in some cases, transliteration mapping can significantly increase the probability of finding desired results. Though transliteration may seem to be an unlikely solution, all three of the problems listed above can be solved through accurate transliteration mapping. The businessman from Boston will find that a Google search for "Beijing Duck" finds results concerning restaurants from China more quickly than a search for "Peking Duck."

If the Beijinger is looking for products from e.g. the shoe company Nike, regardless of whether s/he is searching on Google or Baidu, s/he will find at least five times more results by searching for "Nike" than for its Chinese equivalent, 耐克 (*Nàikè*). The Italian looking for a song written in the International Phonetic Alphabet (IPA) as [ˈkʊt ʃʰˈkʊtʃʰˈhoˌtaːˈhe] will find far more results for "Kuchh Kuchh Hota Hai" than for "Kuch Kuch Hota Hai," even though many of the search results will use the second transliteration.

Such situations reflect a growing barrier in intercultural communication. The web is an important tool in modern communication, and search is the primary application for locating information. However, the web is also a barrier to communication in that a large proportion of web pages are in English. When dealing with foreign terms, such pages generally contain transliterations of foreign terms. There is little standardization of transliteration, thus it is often difficult to find relevant results when search queries involve transliterated terms originally from non-Romanized languages. Search engines can suggest alternate spellings but do not provide alternate romanized transliterations of words from Chinese languages, Japanese, etc. This paper describes a new approach that enhances search engine performance on terms with pronunciations in Mandarin Chinese and Hindustani.

Generally, the user of a search engine could transliterate words from any language in several different ways. However, the original search term may not necessarily be the same as the most commonly accepted transliteration of the word in Roman letters.

Considering Mandarin transliteration, one might transcribe the name of the capital of China (PRC) as Beijing, which is how it would be transliterated in *Hànyǔ Pīnyīn* (unaccented, i.e. without tones). Google searches for the entered word (in some cases, if the term appears to be misspelled, it will also search for a few spelling variations). However, it will not search for transliteration variations. For example, one transliteration of Beijing is Peiching. A Google search for "Beijing" will not find links to Peiching.

Considering Hindustani transliteration, there is a type of bread known in Hindustani (and some other languages) as [nãːn]. Its name may be tranliterated as "nan" or as "naan." Again, Google searches for the entered search term but does not look for transliteration variations. Thus, a search for "nan" will yield more results than "naan," but a search for "naan" will include results concerning the bread more quickly than a search for "nan."

Clearly, detailed replacement patterns are needed to search for alternate transliterations to terms entered by a searcher. Replacement algorithms, in combination with other strategies such as eliminating unlikely search terms, can then be used to search effectively for alternate transliterations of search phrases. This paper analyzes the effectiveness of one replacement algorithm designed to improve search engines.

PREVIOUS WORK

The focus of this paper is transliteration into Roman letters, especially in Mandarin and Hindustani. Hindustani transliteration is a relatively unexplored field, though some research has been done regarding transliteration in "Hindi" and "Urdu" individually. However, many researchers have written about various aspects of Chinese transliteration, in particular:

1. the transliteration of English (or non-Chinese) names into *Hànzì*,
2. the rendering of these names into *Pīnyīn*, and
3. the transformation of Chinese names from *Pīnyīn* to the original *Hànzì*.

Others have focused on similar transliteration problems in other languages, e.g. Arabic and Japanese. [10] mentions an IBM system for transliteration of Arabic names. They then extend this previous algorithm in order to create a program transliterating the Arabic versions of the Western names back into Roman script. Their program deals with an aspect unique to languages using the Hebrew and Arabic scripts: most texts in Arabic (as well as in Hebrew, Persian, Yiddish, Urdu, Pashtu, and many other languages) substantially neglect to write vowels to represent their spoken equivalents.

[10] also mentions a method by Knight and Graehl that probabilistically constructs alternate terms, finally picking optimal search terms through algorithms using finite graphs.

Some other papers have used similar algorithms in order to solve such problems in Japanese and Chinese, i.e. to use accepted transliterations of names common among those who speak the particular language and convert the Romanizations into *kanji* and *hànzì*, respectively. Yet another paper [6] uses a syllable-by-syllable transliteration method in order to transliterate foreign names from Roman letters into *katakana*, a writing system normally used in order to write foreign words in Japanese.

[3] describes how to find transliterated pairs, i.e. terms in languages other than Mandarin and their transliterations into traditional *hànzì*, from the Web. The target of the transliteration is Chinese. It mentions various transliteration systems used for Mandarin (Wade-Giles, *Tōngyòng Pīnyīn*, and *Hànyǔ Pīnyīn*). Terms are segmented from left to right. [11] introduces another left-to-right algorithm focused on transliterating English terms, especially English names, into Chinese. [1] creates Chinese terms out of English terms, such as names. It tries to find matching Chinese characters for English phonemes.

No such methods were found to be used for Hindustani. Much research concerning transliteration in Hindustani is oriented not towards mapping transliterations specifically in Hindustani but towards creating input systems in a set of languages (including Hindustani). [4] and [9] are examples of efforts to create input systems for Hindustani and other languages with similar writing systems. [7] uses N-grams in order to facilitate search in "Hindi" (i.e. Hindustani written in Devanagari script) using Roman script. This technique reduces the problems of Hindustani transliteration described below. However, it does not take into account the variations in Hindustani transliteration that actually exist.

[2] is an algorithm in which search terms are transliterated from *Hànyǔ Pīnyīn* to seven alternative Romanization systems for Mandarin. The algorithm improves searches by extracting parts of search terms using a right-to-left search method. In an earlier version of this method, some terms have been incorrectly transliterated. Therefore, syllabic division is also used to generate alternative transliterations. The algorithm is currently implemented for Mandarin Chinese search terms written in *Pīnyīn*. The program includes a list of transliteration replacement sets, within which the elements are possible transliterations of the same sound. This is the algorithm examined in this paper.

TRANSLITERATION IN MANDARIN

Mapping transliteration systems is not especially problematic in Mandarin, and it can improve search results. Nevertheless, familiarity with Mandarin transliteration systems is essential to mapping them correctly. This section explains how Mandarin is transliterated using Roman script.

Many transliteration systems based on the Roman script exist nowadays for Mandarin, mainly due to political reasons. For example, *Hànyǔ Pīnyīn* is the official transliteration system in the People's Republic of China (PRC). But in the Republic of China (Taiwan), a variation called *Tōngyòng Pīnyīn* is the official transliteration system while its predecessor, Wade-Giles, remains in popular use both inside and outside the country. Most transliteration systems for Mandarin are consistent with one another, and it is not difficult to find a chart that maps most of these systems to one another.

A popular but now unofficial transliteration system, Wade-Giles was originally developed by Herbert Giles in 1892 and is one of the oldest transliteration systems still popularly used for Mandarin. Since no voiced plosives, fricatives, or affricates exist in Mandarin, letters used to indicate these voiced sounds in European languages are not used in Wade-Giles. Aspiration and tone are phonemic in Mandarin, so aspiration is indicated with an apostrophe and tone is indicated with a number. However, many of those

who use Wade-Giles often omit both apostrophes and tone numbers.

One transliteration system based on Wade-Giles is known by various names in English (including "Postal System Pinyin"); in Mandarin, it is usually called *Yóuzhèngshì Pīnyīn*. It was based on Wade-Giles but, like other variants of Wade-Giles, lacked apostrophes or tone marks. It was used primarily in the late 19th and early 20th centuries (or during the late Qing Dynasty) for the purpose of writing addresses. One very widely used example of *Yóuzhèngshì Pīnyīn* is "Peking," which, in some English phrases (e.g. "Peking duck"), has become an almost irreplaceable word.

Gwoyeu Romatzyh, or GR, is another transliteration system developed by Yuen Ren Chao for Mandarin in the early 20th century. The spelling of a syllable in GR varied depending on its tone. For instance, what would be written in *Pīnyīn* as *mā* would be written in GR as *mha*, while *má* in *Pīnyīn* would correspond to *ma* in GR. Though GR is no longer in widespread use, it influenced later transliteration systems (including *Pīnyīn*), which borrowed its innovation of using Roman letters that usually represent voiced phonemes to represent voiceless unaspirated phonemes in Chinese (e.g. b for [p]) and letters usually representing voiceless phonemes to represent voiceless aspirated phonemes (e.g. p for [p^h]).

Later, during the Second World War, Yale University designed a system called Yale Romanization, which, to this day, is sometimes considered a relatively practical model of Mandarin pronunciation from an American point of view. Like Wade-Giles, it did not have one transcription per tone. Rather, it used diacritics to indicate the tone of a word; *Pīnyīn* also includes these diacritics. Yale Romanization is still used in some books, though *Pīnyīn* is now used much more frequently.

The transliteration system that has been officially used for Mandarin in Taiwan since 2000 is a variation of *Hànyǔ Pīnyīn* called *Tōngyòng Pīnyīn*. *Hànyǔ Pīnyīn* and *Tōngyòng Pīnyīn* have only a few differences, since the latter is based on the former. Both share many similarities with Yale Romanization, but *Hànyǔ Pīnyīn* includes e.g. the use of *q* to represent the sound [$tɕ^h$] (compare *ch* in Yale Romanization and *c* in *Tōngyòng Pīnyīn*).

Almost all of these transliteration systems, with the exception of *Yóuzhèngshì Pīnyīn*, are easily mapped to one another. Thus, it is possible to convert search terms written in one transliteration system to several other transliteration systems and search for all results. Nevertheless, search engines like Google do not yet use this advantage of Chinese transliteration to facilitate search.

TRANSLITERATION IN HINDUSTANI

If mapping Mandarin transliteration systems for the purpose of intercultural web search is a simple task, mapping Hindustani transliterations is much harder. This means that searching for Hindustani words in Roman script is also difficult, and it is another problem that search engines have not yet solved.

Hindustani is the most widely spoken language in India, and films in Hindustani are very popular both in India and worldwide. Songs from these films are popular among Indians all over the world, and many of them are available online. However, their titles are usually transliterated in Roman script. Searching for these songs can be a difficult task due to the nature of transliteration in Hindustani.

Unlike Mandarin, which has several official transliteration systems with *Pīnyīn* becoming the most widespread, Hindustani has no widely used official transliteration systems. In addition, Hindustani is diglossic, i.e. written in two scripts. As a result, the problems with transliteration in Hindustani are much more difficult to solve.

Hindustani may be written either in the Sanskrit-based Devanagari script or in the Persian-based Nasta'liq script. When written in Devanagari, it is called Hindi; when written in Nasta'liq, it is called Urdu. There are some differences in terms of the number of sounds that are represented in each script, as well as the number of letters per sound. For example, Nasta'liq includes distinct symbols for the sounds [q] and [x], which are borrowed from Arabic and Persian. Devanagari does not, though these sounds may be indicated by including a dot (or *nuktaa*) with the letters *ka* and *kha*, respectively. Conversely, Devanagari includes two symbols for the phoneme [ʃ], i.e. श and ष (both pronounced [ʃə]), which originally represented two distinct sounds. Nasta'liq includes only one symbol, i.e. the letter *sheen* or ش.

This means that words in Hindustani may be transliterated in any of the following ways:

(a) in accordance with how they are written in Devanagari,

(b) in accordance with how they are written in Nasta'liq,

(c) in accordance with both, or

(d) in accordance with neither.

Transliteration can be influenced by script, but few transliteration systems are consistent with both Devanagari or Nastaliq. Some transliteration systems for Hindustani are focused towards either writing system. Most often, however, Hindustani when written in Roman script is consistent with neither script. In fact, transliteration systems are usually created on an individual basis and inconsistently followed. It may often be easy for a human to understand Hindustani text written in Roman script, but to make a

computer compare transliterations of Hindustani is a much more difficult task.

TRANSLITERATION ALGORITHM

The challenges posed by transliterations of Mandarin and Hindustani for intercultural web search are quite different in nature. However, no solution had been proposed for either of these challenges before the algorithm described in [2]. Though this algorithm is specifically designed for mapping transliterated Mandarin search terms, it may be applied to transliterated Hindustani as well. The approach depends on the use of a table to map transliteration systems. The table is organized so that each column corresponds to a particular transliteration system, while each row corresponds to a particular sound (e.g. phoneme, syllable).

The table used in the algorithm is a two-dimensional array. The first column is used as a base transliteration system. As more transliteration systems are implemented, the number of columns and (to a lesser extent) rows will increase. However, the final number of columns and rows is finite.

Algorithm

The algorithm used in [2] uses the table for a procedure which may be summarized as follows: Break up the search term into parts. Convert each of these parts into their equivalents in other transliteration systems. Finally, for each transliteration system, put the parts back together and search for the transliterated term. This method is described in greater detail in the following paragraphs.

For the decomposition and composition algorithms below, let Patterns be a two-dimensional array consisting of rows and columns. In the following, Patterns [r][c] indicates the string in column c of row r. The word "searchterm" in the decomposition algorithm is initially the text entered by the user.

This is the decomposition algorithm:

1. Let "query" be searchterm. Proceed to step 2.
2. Is query equal to column 0 of any row in Patterns, i.e Patterns [row][0]? If so, store this row number in a list called Parts. Then proceed to step 4. If not, proceed to step 3.
3. Remove the last letter from query. Proceed to step 2.
4. Remove query from the beginning of searchterm. If searchterm is empty, proceed to step 5. Otherwise, proceed to step 1.
5. The decomposition is complete; Parts contains the row numbers of the decomposition.

This is followed by a composition algorithm:

1. Let column be 1. Proceed to step 2.
2. For each row in Parts, find the string in Patterns [row][column]. Proceed to step 3.
3. Append the strings found in step 2 to form one string called "newterm." Proceed to step 4.
4. Put newterm in a list called Terms if newterm is not already in this list. Proceed to step 5.
5. Increase column by 1. If column is greater than 7, then stop. Otherwise, proceed to step 2.

The original search term, along with the strings that are now in Terms, is the output of the algorithm. Google is used to search for all of these terms (using a facility available via Google allowing programs to search the web). The program then eliminates any combinations that do not produce a significant number of results.

This algorithm was first proposed in [2]; it is somewhat similar to the Lempel Ziv Welch compression algorithm (LZW). LZW builds a table of parts as it compresses strings and matches longest strings. LZW however processes strings from left to right, looking up longest matches using some ordering on patterns. The algorithm proposed in [2] uses the right to left scanning to avoid having to store patterns in any particular order. It is more inefficient than LZW in string decomposition, but this is not an issue in decomposing humanly entered search terms.

Application to Mandarin

The method described here depends on knowing the relationships between different transliteration schemes such as Wade-Giles, Yale transcription, etc. The current version of the algorithm only takes into account a few variations of *Hànyǔ Pīnyīn* (indicated by HP, MHP1, and MHP2), *Tōngyòng Pīnyīn* (TP1 and TP2), and Wade-Giles (WG1, WG2, and WG3).

This information is stored in simple tables in the current version. The program uses tables of replacement patterns within the algorithm described in the last section. The table used in the algorithm for Mandarin is a two-dimensional array consisting of 8 columns and 110 rows.

The information about alternate transliterations is organized into an array divided into rows and columns. A match on a term inside a particular sub-array, such as {"üe", "yue", "yue", "ue", "ve", "üeh", "üeh", "üeh"}, means that alternate transliterations within the same array may be used to replace that particular matched part.

The algorithm was tested using 130 *Pīnyīn* search terms consisting of Chinese city & province names. For 16 of these terms, no other transliteration was found. 60 terms had at least two other transliterations, and six had three more transliterations. Of the search results, approximately 93% were found by searching for the original *Pīnyīn* term. This process was completed in five minutes and twelve seconds on a 2.8-MHz AMD PC with a cable modem Internet connection.

One relatively minor complication involved in transliterating Mandarin search terms is the necessity of syllabic division. Without syllabic division, some search terms in *Pīnyīn* were originally transliterated incorrectly. With more transliteration systems, more such search terms could be erroneously transliterated. An adjustment to the original algorithm was therefore necessary in order to increase the range of search terms that could be transliterated.

Application to Hindustani

It may be possible to apply the technique of the algorithm to Hindustani. However, if this is to be done effectively, the problem of inconsistency (see "Transliteration in Hindustani") must be solved. No solution has yet been found for this problem, but the basis of an attempted application of the algorithm to Hindustani is presented below.

A base transliteration system has been invented to be consistent with both Devanagari and Nasta'liq. Unfortunately, it is very case-sensitive, and the need to differentiate between letters that occur in Devanagari and similar letters in Nasta'liq results in some irregular symbols (e.g. "nx" for a nasalized vowel). Transliterations of Hindustani may also use punctuation marks; for example, [8] uses "n." (i.e. n + period) to indicate a nasalized vowel.

The table used currently consists of 6 columns and 67 rows. The creation of an additional 7 columns has begun, but none of these additional columns is complete. As in the original algorithm, the number of rows and columns is flexible but finite. The basic method used is also the same. Transliteration information is based on all available written transliterations of Hindustani in Roman script.

The first five rows of the transliteration tables used for Hindustani are provided below:

a	a	a	a	a	a
aa	aa	ā	aa	ā	aa
i	i	i	i	i	i
ee	ee	ī	ee	ī	ee
u	u	u	u	u	u

However, transliteration systems in Hindustani are comparable to *Yóuzhèngshì Pīnyīn*, so this technique will work only for the relatively few transliteration systems for Hindustani that actually are consistent. In fact, unlike *Yóuzhèngshì Pīnyīn*, even the same word may be written in multiple ways in the same source. For this reason, adjustments are needed to improve the effectiveness of the algorithm as applied to Hindustani.

FURTHER RESEARCH

There are a number of limitations that may be improved in the future. One comparatively minor problem is the low number of transliteration systems included for Mandarin and Hindustani. This problem can be solved by simply making another column for any missing transliteration system and providing the system's equivalents for the terms in the first column.

Due to the increased usage of computers and popularity of keyboards using the Roman alphabet worldwide, this paper has focused on transliteration involving the Roman alphabet. Nevertheless, transliteration between scripts of different languages is also a common problem in intercultural situations, because people need to express the sounds of one language using the script of another language.

Another difficult problem is to generalize the algorithm to other non-English languages, particularly other East, South, and Southeast Asian languages. A different set of component replacements would be needed for each language, and since each language deals with transliteration differently, the approach may have to be modified as well. The situations of transliteration in Tibetan, Romani, Malayalam, and Japanese are detailed below, followed by possible approaches to the problem of applying the algorithm to these languages.

The main complication in transliterating Tibetan in Roman script is the difference between how words in Tibetan are spelled and how they are pronounced. The Tibetan alphabet includes many symbols that may not correspond to their phonetic equivalents in Modern Tibetan. One widely accepted transcription, known as Wylie, transliterates Tibetan words based on the orthography (i.e. how Tibetan words are spelled). Several others, however, transliterate the same words based on sound. Thus, Wylie will be adopted as the base transliteration system, since it is a generally accepted romanization system and is consistent with the orthography. There appear to be no other systems based on the orthography; however, there are many transliteration systems that are based on (and often consistent with) the phonetic pronounciation and will be documented.

When applying the algorithm to Romani, the main issue to take into account is the lack of literate speakers. In Romani, transliteration generally varies depending on the country of origin of the literate speaker. For this reason, the word for "to be able" (pronounced [ʃaɪ]) is written in various ways, e.g. *šaj* (among some literate speakers of the Vlax Dialect),

shay (among some other literate Romani-speakers), *shy* (among some literate Romanies in England), and *schaj* (among some literate Romanies in German-speaking regions). Thus, those searching for a search term in Romani might use any one of several such unofficial transliteration systems. A modification of this algorithm specifically designed for Romani would be capable of searching for any of these transliterations.

It is also possible to apply this method to Malayalam with some adjustments. There are few transliteration systems that have been officially devised for Malayalam. In Malayalam, as in Hindustani and many other Indian languages, words are generally transcribed depending on how the user believes the word in question should be spelled. Thus, an input system could be used as a base transliteration system for Malayalam, but (as in the case of Hindustani) some adjustments may be needed to this algorithm.

Finally, the algorithm can be used to map Japanese transliterations with relative ease. In terms of difficulties concerning transliteration systems, Japanese resembles Mandarin Chinese. Japanese speakers tend to use Romanization, or *Rōmaji* in Japanese, quite systematically when transliterating Japanese words.

The most widely used Romanization system is the Hepburn system (sometimes with slight variations), but the Kunrei and Nippon systems are also somewhat common. Other transliteration systems are occasionally used for purposes of creativity. However, perhaps the greatest deviations from these three systems may be found in transliteration systems used by non-Japanese. For example, non-Japanese media are generally more likely than Japanese media to ignore the phonemic differences between long and short vowels. Hence, “sayonara” is a widely used transliteration (usually used by non-Japanese) for the Japanese word for “goodbye,” though the “o” is a long vowel. The same word would be written *sayōnara* in the Hepburn system and *sayônara* in the Kunrei and Nippon systems.

In an application of this algorithm to Japanese, the Hepburn system could be used as a base system to be mapped to all other systems. The other systems will include other variations of the Hepburn system and the Kunrei and Nippon systems. Less widely used and less systematic transliteration systems will also be included. The base system will consist of individual vowels and consonants used in Japanese. Thus, for example, instead of sets "sa, shi, su, se, so" there will be sets for "a, i, u, e, o, s, sh."

Generally, if this algorithm is applied to any two languages, the version for one language will be quite distinct from the other version. For this reason, transliteration knowledge is necessary to implement the algorithm in any given language.

CONCLUSION

Web search for romanized transliterations of words from non-Romanized languages can be improved to facilitate intercultural communication. In this study, the effects of a proposed method for searching for words of Mandarin and Hindustani origin are examined. The method is found to be a more effective improvement for some languages than for others, but it may be possible to extend it to more languages with some further developments. Using the method referenced in this paper, search engines could improve their facilities for those searching for any information concerning non-Western cultures.

REFERENCES

[1] Gao Wei. Phoneme-Based Statistical Transliteration of Foreign Names for OOV Problem. Master's Thesis (June 2004), The Chinese University of Hong Kong.

[2] John, Vijay. A Method for Enhancing Search Using Transliteration of Mandarin Chinese. Texas Linguistics Society 10 (2006). University of Texas. http://uts.cc.utexas.edu/~tls/2006tls/papers/john_tlsx.pdf.

[3] Kuo, Jin-Shea and Ying-Kuei Yan. Generating Paired Transliterated-Cognates Using Multiple Pronunciation Characteristics from Web Corpora. *PACLIC 18* (Tokyo, December 8th-10th, 2004), Waseda University, pp. 275-282.

[4] Madhavi, Ganapathiraju, Balakrishnan Mini, Balakrishnan N., and Reddy Raj. Om: One tool for many (Indian) languages. *Journal of Zhejiang University SCIENCE* (Hangzhou, China, September 10, 2005), Zhejiang University Press.

[5] Meng, Helen, Berlin Chen, Erika Grams, Sanjeev Khudanpur, Wai-Kit Lo, Gina-Anne Levow, Douglas Oard, Patrick Schone, Karen Tang, Hsin-Min Wang, and Jian Qiang Wang. Mandarin-English Information (MEI): Investigating Translingual Speech Retrieval. University of Pennsylvania, October 2000.

[6] Matt Mettler. TRW Japanese Fast Data Finder. TRW Systems Development Division, Redondo Beach, California.

[7] Natrajan, Anand, Allison L. Powell, and James C. French. Using N-grams to Process Hindi Queries with Transliteration Variations. University of Virginia, 1997.

[8] Prabhu, Dinesh. Urdu Dictionary. http://smriti.com/urdu/urdu.dictionary.html.

[9] Raghavendra, E. Veera, Prahallad Lavanya, and Fahmy Mostafa. Transliteration editors for Arabic, Persian, and Urdu. *Proceedings of International Conference in Universal Digital Libraries* (Pittsburgh, 2007).

[10] Stalls, Bonnie G. and Kevin Knight. Translating Names and Technical Terms in Arabic Text. COLING/ACL Workshop on Computational Approaches to Semitic Languages (Montreal, Quebéc, 1998).

[11] Wan, Stephen and Cornelia Maria Verspoor. Automatic English-Chinese Name Transliteration for Development of Multilingual Resources. Microsoft Research Institute (Sydney, Australia). Macquarie University

[12] Wu, Jian-Cheng, Tracy Lin, and Jason S. Chang. Learning Source-Target Surface Patterns for Web-Based Terminology Translation. *Proceedings of the ACL Interactive Poster and Demonstration Sessions* (Ann Arbor, June 2005), pp. 37-40.