An Improved Method for Enhancing Search Using Mandarin Transliteration and Syllabic Division

VIJAY JOHN

1 Abstract

This paper deals with a relatively unexplored aspect (searching for romanized transliterations of terms) of a popular language (namely, Mandarin Chinese). Although search engines are able to suggest alternate spellings, they do not yet look for all possible transliterations of the same word. This paper describes a new approach to enhance search engine performance on terms originally from Mandarin Chinese. 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. This algorithm, Xiăozhī, 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. Even though the implementation is specific to Mandarin Chinese, Xiăozhī may be extended to other languages using a different list of sets. This includes less-studied languages, such as Tibetan, Romani, and Malayalam.
2 Introduction

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. Our proposed algorithm can expand searches to include relevant transliterations of Mandarin Chinese search terms in addition to the original Hán yǔ Pīnyīn (unaccented, i.e. without tones)\(^1\) search terms.

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.\(^2\)

For example, one might transcribe the name of the capital of China (PRC) as Beijing. 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 will not find links to Peiching, but a search using Xiāozhī (which in fact are links to the “City of Peiching” which in fact are links to the same capital of China).

For romanized transliterations of Mandarin words, this problem of alternate transliterations is complicated. Mandarin’s official script is either Simplified or Traditional Hán zì ("Chinese characters").\(^3\) There are not one

\(^1\)In this paper, the word Pīnyīn (when used in isolation) refers specifically to Hán yǔ Pīnyīn. In Mandarin, Pīnyīn literally means "spelling sounds" and is normally used to refer to other transliteration systems as well (including Wade-Giles).

\(^2\)Note that, unlike a standardized transliteration, "the most commonly accepted transliteration" does not necessarily have to be accepted in all countries. For example, in the case of Mandarin, Hán yǔ Pīnyīn is the official transliteration system of the People's Republic of China, which is the most populated country in the world. Therefore, we will consider Hán yǔ Pīnyīn to be "the most commonly accepted transliteration." However, Taiwan officially uses Tōngyǒng Pīnyīn, a system formerly not implemented in Xiāozhī.

\(^3\)Simplified characters are officially used in the People's Republic of China and in Singapore whereas traditional characters are the official script in the Republic of China (Taiwan). Other languages that use hàn zì to a slightly limited extent, such as Japanese (in which they are known
but several official and reputed transliteration systems that exist for the language, depending on the particular country. Other well-known entities (e.g. Yale University, Herbert Giles's A Chinese-English Dictionary) devise phonetic transliteration systems and thus introduce even more variations. In addition, searchers from different cultural and social backgrounds often misspell words and transcribe words in very different ways.

Furthermore, even accepted transliteration systems for Mandarin Chinese may have significantly different spellings not only of single phonemes (as in English) but also of entire syllables that are between one and four Roman characters long. For example, Mandarin syllables that are spelled zhun in Hányù Pīnyīn proper would be written as chun in Wade-Giles (a widely used transliteration system in Taiwan), jwun in Yale Transcription, and juen in Gwoyeu Romatzyh (formerly the official transliteration system in Taiwan).

Clearly, detailed tables of 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 proposes one such algorithm that has the potential to improve search engines.

Though this particular version of Xiăozhì deals with Mandarin Chinese and is designed mainly for search engine use, it could also be applied to research archives of samples in lesser studied languages. Data in research archives may be transcribed using a variety of different transliteration systems. This algorithm could be used to find several transliterations of the same word in various archived documents and thereby facilitate archive search.

In addition, if the archived data is transcribed using the IPA system, it is possible that the same word is nevertheless transcribed in more than one way due to misconstruction. The method described below may therefore be applicable even if all data within a linguistic archive is transcribed in IPA.

as kanji), Korean (which uses the term hanja), and other Chinese languages (Cantonese, Wú, Mín languages, Gan, etc.) sometimes use a combination of both. However, they tend to adopt traditional characters, because in such areas, hánzǐ has often been used for hundreds of years. (The simplified characters have only been in existence since the 20th century.)
A prior version of Xiăozhī was developed before March 2006. At the time, however, it did not include a table with columns organized by transliteration system. Only four of the numerous systems used to create the sets in the previous version are employed in the current edition.

3 Previous Work

The focus of this paper is transliteration of Chinese words into Roman letters. Many researchers have written about various other aspects of Chinese transliteration, in particular:

1. the transliteration of English (or non-Chinese) names into Hànzī,
2. the rendering of these names into Hányū Pīnyīn (Mainland China's Romanization system), and
3. the transformation of Chinese names from Pīnyīn to the original Hánzī.

Still others have focused on similar transliteration problems in other languages, e.g. Arabic and Japanese. Their techniques are more similar to the algorithm described below than to Xiăozhī. [Stalls and Knight] mention 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.

[Stalls and Knight] also mention a method by Knight and Graehl that probabilistically constructs alternate terms, finally picking optimal search terms through graph algorithms.

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 hánzī, respectively. Yet another paper [Mettler] 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.

---

4More specifically: [Wu], [Gao], [Kuo], [Meng], and [Wan]
[Kuo] describes how to find transliterated pairs, i.e. terms in languages other than Mandarin and their transliterations into traditional hànzi, from the Web. The target of the transliteration is Chinese. It mentions various transliteration systems (Wade-Giles, Tōngyòng Pīnyīn, and Hányù Pīnyīn). Terms are segmented from left to right. [Wan] introduces another left-to-right algorithm focused on transliterating English terms, especially English names, into Chinese. [Gao] creates Chinese terms out of English terms, such as names. It tries to find matching Chinese characters for English phonemes.

[Google] is part of the help section of the Simplified Chinese version of Google. It allows the searcher to use a slightly modified version of Hányù Pīnyīn (not yet included in Xiăozhì) to search for Chinese characters. If a term in Pīnyīn is entered, Google will search only for the original search term. Towards the top of the page displaying search results, it will also provide Chinese characters that could be spelled similarly in Pīnyīn (or a variation thereof). However, if the user wishes to search for the hànzi transcribed as lǚ, lüe (or lue), nǚ, nüe (or nue), then "lv, lve, nv, nve," respectively, must be entered as the search term. Otherwise, Google will simply search for the original search term. Like the references mentioned here, Google does not address the problem of enhancing search in romanized letters of terms from Mandarin, although it deals with transliterations.

4 Initial Program

My initial idea was to create a program utilizing a long list of transliterated sets used to replace one transliteration with another. A transliterated set is a set of strings. For example, \{zhun, chun, jwen, juen\} is a transliterated set.

The program was originally designed to operate in the following manner: it was assumed that the original search term and the first element in each set were written in Pīnyīn without any tones. The other elements in each set reflected both relatively accepted systems of transliteration and various cultural backgrounds. For example, "iaow" is also written as "yau" (Yale Transcription) and "iaor" (to acknowledge the tendency of Mandarin speakers from Beijing to add "r" to the end of many words, e.g. diànyīngr for "movie theater").

This computer program was written in Java. In a Command window, the searcher could type in a term in unaccented Pīnyīn. The program would,
in turn, search Google for all results of the original term and of a modified term. However, a major flaw was soon found with this approach: the program would make several unnecessary changes before searching for just one alternate transliteration. The term "ying" was, according to the list, to be transliterated only as "ying" and "yingr." Instead, the program changed "ying" to "yink" to "yenk" to "yemk" to "yermk" to "yarmk,'" and searched for "yarmk,'" as well as "ying." As a result, Google ultimately searched only for results containing "ying" and those concerning the Yarmuk River in the Middle East. The program searched in this manner because of the following reasons:

1. "g" is transliterated as "k" in Wade-Giles;
2. "Pīnyīn k" is transliterated as "k'" (k with an apostrophe) in Wade-Giles;
3. "i" may be transliterated as "e," due to similarities in pronunciation in many Mandarin words (for example, sè (color) and sì (four), both formally pronounced as [s_] with a falling tone);
4. "n" might be confused with the letter "m" due to their strong similarity, proximity on the keyboard, nasal quality, etc.,
5. "e" in Beijing dialect is pronounced "er,"
6. "er" might be confused with "ar," since Mandarin-speakers may pronounce both sounds similarly (for example, the number "two," written èr but often pronounced [a:tax] with a falling tone). 

In summary, the problems were the tendency to make replacements from many transliteration systems and an improper string search method.

5 Proposed Algorithm

The problem would have been relatively less grave, I found, if the program had used the original Pīnyīn search term to search the list for individual components (e.g. ying) rather than individual letters (e.g. y, i, n, g). The program was changing each result letter by letter. Furthermore, the conventional left-to-right scanning technique used in the program resulted in poor matches and depended on the order in which patterns were arranged. Another improvement that would increase the program's effectiveness in transliterating, I later realized, would be a table to organize all of the columns (not just the first) so that each column uses a particular transliteration system.

The table used in Xiūzhī is a two-dimensional array consisting of 8 columns and 110 rows. 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. This is because each column
represents one transliteration system, and each row represents one component.

More details on how the array (a key ingredient to the effectiveness of the algorithm) is created can be found in Section 6 of this paper.

For the decomposition and composition algorithms below, let Patterns be a two-dimensional array consisting of 8 columns and 110 rows. 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 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 Xiăozhì. I use Google to search for all of these terms (using a facility available via Google allowing programs to search the web). The
This approach is compatible even with most multisyllable words, although the program does not yet include a system indicating the division of syllables. For example, when the word pengren (pēngrèn in Mandarin Chinese means "cooking" but is composed of two syllables with distinct meanings: pēng-rèn) is entered into the system, the results should be determined by looking first for "pengren," then "pengr," etc. until it finds the true components: p, eng, r, en. Replacements can then be done by substituting each of these components with another string in its set. The sets in this example are:

- \{"i", "i", "i", "i", "i", "i", "i"\}
- \{"eng", "eng", "eng", "eng", "eng", "eng", "eng"\}
- \{"en", "en", "en", "en", "en", "en", "en"\}

Here p is replaced with p and p'. Similarly, the program replaces r with r and j. Resultantly, the corresponding search patterns are obtained: pengren (any system used other than Wade-Giles) and p'engjen (any Wade-Giles system). Though more than one transliteration system shares the same search pattern, the program discards all repetitions of search patterns. Thus, the number of search patterns created may be fewer than the number of transliteration systems, regardless of the number of syllables involved.

Speed and efficiency are not issues in Xiāozhī due to the small, manageable number of computational steps involved. Let K be the number of rows in the array (currently K = 110). If query has N characters, then step 1 of the decomposition algorithm will be performed N or fewer times. The total number of computational steps involved in step 1 is therefore a simple arithmetic series, i.e. at most K(N^2 + N) / 2, as shown in the following paragraph.

If the length of the query in Step 1 is M, step 2 involves a maximum of K computational steps (i.e. one computational step per row searched). Step 3 can be performed at most M times. As a result, steps 2-3 of the decomposition will involve at most KM computational steps. Initially, M is equal to

---

5The program determines how many results are "significant" by comparing the number of results for all combinations. For example, if "ding" yields five billion results and "ting" yields only two results, "ting" has not produced a "significant" number of results. Therefore, "ting" is eliminated.
N. The next time through step 1, M is equal to $N - 1$, etc. The total number of steps, then, is a simple arithmetic series $K(N + N - 1 + N - 2 + \ldots + 1)$, i.e. at most $K(N^2 + N) / 2$.

Since $K$ is 110, and $N$ is typically at most 50, $K(N^2 + N) / 2 < 150,000$. This is a small number of steps for most computers. Since 8 transliteration systems (and thus 8 columns) are currently included in the array, Google is queried at most 8 times. For this reason, Xiāozhī may take a few seconds to produce results. Overall, however, the speed of the algorithm’s performance is of little or no practical concern.

I have not found the exact same algorithm as Xiāozhī anywhere, however there is some correspondence between Xiāozhī and 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. Xiāozhī uses the right to left scanning to avoid having to store patterns in any particular order. Xiāozhī is more inefficient than LZW in string decomposition, but as I have described in the last paragraph, this is not an issue in decomposing humanly entered search terms.

6 Adding Transliteration Knowledge

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 Hányū 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 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 first term is written in the HP or Hányū Pīnyīn system. The second term is TP1, i.e. Tōngyòng Pīnyīn System #1. The third is TP2, or Tōngyòng Pīnyīn System #2. The fourth is MHP1 (Modified Hányū Pīnyīn System #1), and the fifth is MHP2. The last three are WG1, WG2, and WG3.)
Wade-Giles transliterations have been included into the program mainly based on a table provided in [Chen]. These tables consist of all syllables (without tones) that exist in Mandarin Chinese. They are organized alphabetically using the Yale syllabary (which, in the current version of Xiăozhî, is not yet included).

The specific information in some parts of the table has been included in separate sets. Thus, all information pertaining to the unaccented Pinyin syllable quan has been included under set quan, i.e. the set beginning with quan. Generally, however, information pertaining to the transliteration of one syllable has been decomposed into elements of the sets created in the program. For example, transliterations for standard Pinyin syllable lue include luye (Tongyong Pinyin Systems #1-2), lve (Modified Pinyin #2), and lüeh (Wade-Giles). For this reason, yue (columns TP1 and TP2), ve (MP2), and üeh (WG1, WG2, and WG3) have been entered as elements in set ue.

Xiăozhî was tested using 130 Pinyin 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 Pinyin term. This process was completed in five minutes and twelve seconds on a 2.8-MHz AMD PC with a cable modem Internet connection.

The implementation of the algorithm is available from the author. It consists of Java source code that requires a license from Google to utilize Google’s API for performing the searches (after creating alternate transliterations using Xiăozhî). Xiăozhî can be tested online through a link at http://www.translitsearch.com/demos/demos.htm.

7 Improvements in Effectiveness

Nevertheless, there are some search terms that Xiăozhî has failed to map accurately. One constraint of standard Hanyû Pinyin prevents the transcription of almost any syllable in Mandarin from beginning with the vowels i or u. However, Xiăozhî formerly decomposed some polysyllabic Pinyin search terms so that one syllable of the term began with either of the two vowels.

One example of such terms is the Mandarin word written in Pinyin as zhúniân, which means "year after year." One of the sets in the list begins
with the element zhun, so the program would originally have searched for zhun and ian in the list of sets. However, since ian cannot be a syllable in Pinyin, this search is inappropriate. In reality, the word zhunián is composed of two syllables: zhú and nián. The correct components are zh, u, n, and ian.

Incorrectly decomposed search terms of this type include the letters i or u and the following features, hereafter known as prerequisites:

1. A vowel after the i or u, as well as
2. A vowel or n before the i or u, and perhaps
3. The combination ng immediately preceding the u.

Currently, an addition to the previous algorithm generates more accurate transliterations by separating the letter preceding the i or u from the previous component. This is done via another function that refers to the original decompose function (see Section 5):

1. Does searchterm include the vowels "i" and/or "u"? If so, proceed to step 2. If not, decompose query.
2. Does searchterm include a vowel immediately after an "i" or "u"? If so, let "mid" be the "i" or "u" followed by another vowel. Then proceed to step 3. If not, decompose query.
3. Does searchterm include a vowel or an "n" immediately before mid? If so, proceed to step 5. If not, proceed to step 4.
4. Is mid immediately preceded by "ng"? If so, proceed to step 5. If not, decompose query.
5. Let "query" be searchterm up to the vowel or "n(g)" immediately before mid. Proceed to step 6.
8. Decompose searchterm.

In some cases, Xiăozhī could transliterate certain words that included the prerequisites correctly. For example, the term zhunian is correctly transliterated, though the components are erroneously generated. Zhunian would be transliterated as chunien in Wade-Giles regardless of whether the decomposition is zhun + ian or zh + u + n + ian.

In Pinyin, neither ng nor g immediately precedes i.
It is mainly because of the small number of transliteration systems included that the term was correctly transliterated. Other systems distinguish between \(zh + u + n\) and \(zhun\); they would thus produce incorrect results. For instance, if Yale Transcription were included in the tables of Xiăozhī, Pīnyīn zhunian would be incorrectly transliterated as \(*j\!wun\!yan\) in Yale Transcription (instead of \(junyan\), i.e. \(j + u + n + yan\)). The syllable \([\text{]|u}\), transliterated as \(zhu\) in Pīnyīn, is transliterated as \(ju\) in Yale Transcription. The syllable \(zhun\) \([\text{]|Y)n\], in contrast, is spelled \(j\!wun\) in Yale Transcription.

Although only nine transliteration systems (i.e. three transliteration systems and variations thereof) are included in Xiăozhī, some words were transliterated incorrectly. The Pīnyīn word binguan is incorrectly transliterated into Wade-Giles as \(*p\!ing\!uan\) and not as \(p\!inkuan\), because Xiăozhī classifies the "g" in this word as a part of the component \(\text{ing}\). In reality, the word is divided into the two syllables \(\text{bin}-\text{guan}\). Because the component "g" is transliterated as "k" in Wade-Giles, the original search term should be changed to \(p\!inkuan\).

One relatively minor complication involved in transliterating Mandarin search terms is the necessity of syllabic division. Thus, some search terms in Pīnyīn including the letters \(-ni-\) or \(-n(g)u-\) were originally transliterated incorrectly. With more transliteration systems, more such search terms could be erroneously transliterated. An adjustment to Xiăozhī was therefore necessary in order to increase the range of search terms that could be transliterated.

8 Future Research

There are a number of limitations that may be improved in the future. One comparatively minor problem is the fact that only three transliteration systems are used in the current version of Xiăozhī. This problem can be solved by simply making another column for any missing transliteration system, providing the system's equivalents for the Pīnyīn terms in the first column, and adding new rows if necessary.

A previous version of this paper included a larger variety of transliteration systems with less organization. The transliteration sets used in this paper need to be expanded to include other accepted transliteration systems (e.g. Yale and Gwoyeu Romatzyh), accents of Mandarin (e.g. the Beijing and Taiwanese accents), and pronunciation of \(h\!\text{anzi}\) in other Chinese dialects (e.g. Cantonese) and foreign languages (e.g. Japanese, Korean).
One important transliteration system that should be added is one known in English as "Postal System Pinyin." This effort is currently hampered by a lack of reliable information concerning this system.

*Pīnyīn* includes some other constraints concerning syllabic division so that there are, even in the current version, some potential search terms that would be incorrectly decomposed. In *Pīnyīn*, every syllable must begin with a consonant unless it is at the beginning of a word (e.g. *Àomén* "Macao") or preceded by an apostrophe or hyphen⁷ (e.g. *pīng’ān* "safe, peaceful"). This means that a word including *n* in the middle of a word must be divided so that one syllable begins with *n*. For example, *Jínán* (a provincial capital) is composed of the syllables *Ji* and *nán*, but *Xiāožī* would currently decompose the syllables *Jīn* and *ān* if given a searchterm *Jīnán*. More improvements are needed so that search terms are divided into the correct syllables, which could then be decomposed.

The transliteration knowledge described in Section 6 is only an introduction of the sources and methods used to construct transliteration tables. The construction of more extensive transliteration tables is a separate topic that needs further investigation.

It is possible 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. The right-to-left replacement method can also have other applications such as stemming and matching. This is further discussed in the next section.

### 9 Applicability to Lesser Studied Languages

Although *Xiāožī* itself deals strictly with terms of Chinese origin, the algorithm can be applied to relatively unknown languages such as Tibetan, Romani, and Malayalam. However, each of these languages deals with transliteration differently.

The Tibetan alphabet includes many symbols that may not correspond to their phonetic equivalents in Modern Tibetan and Dzongkha. The word

---

⁷Hyphens appear to be less commonly used than apostrophes in *Pīnyīn*. Many texts including *Pīnyīn* do not include hyphens where they could be included. For example, [*Pīnyīn*] spells the Mandarin equivalent of the English word "first" as *dìyī*, but the hyphen is often either replaced by a space or simply omitted.
for "hello" in Tibetan is pronounced ta-shi de-leh but written with letters corresponding directly to bkra shis bde legs. One widely accepted transcription, known as Wylie, transliterates Tibetan words based on the orthography. (Thus, the Tibetan word for "hello" would be Romanized as bkra shis bde legs in Wylie). Several others, however, transliterate the same words based on sound. For example, the first syllable in the above-mentioned phrase is transliterated in Tibetan Pinyin (the PRC's officially adopted transcription for Tibetan) as zha.

Another version of Xiăozhī will be created to handle such transliteration issues in the Tibetan language. 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 pronunciation and will be documented.

Xiăozhī can also be applied to Romani, a lesser-studied language in which transliteration generally varies depending on the country of origin of the literate speaker. For this reason, the word for "to be able" (pronounced [jɑːj]) is written in various ways, e.g. šaj (among some literate speakers of the Vlax Dialect), 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 Xiăozhī specifically designed for Romani would be capable of searching for any of these transliterations.

The Vlax Dialect of Romani has a "standard" transliteration system that generally uses letters found on a standard keyboard. However, a few letters in the Vlax Romani alphabet include a háček, which is comparatively difficult to include on a computer. This, too, might cause users to adopt alternative transliteration systems for the purposes of search. (A user familiar with the spelling šaj, for example, might spell the word as shaj for the purposes of using a search engine). Xiăozhī could also include any additional Romani transliteration systems that have been formed for this reason.

Finally, it is possible to apply this method to Malayalam with some adjustments. There are few transliteration systems, if any, that have been officially devised for Malayalam. Malayalam words, like their counterparts in many other Indian languages, are generally transcribed depending on how the user believes the word in question should be spelled. For this reason,
some technique must be added to the Malayalam version of Xiăozhī in order
to determine all possible spellings of words in Malayalam. (It may be ne-
cessary to include a statistical approach and/or a set of rules).

There are some transliteration systems for Malayalam, however, that are
quite consistent with the Malayalam alphabet. These very similar systems
are used in script conversion programs such as Maya and Varamozhi. The
system used in Maya does not require any special diacritic marks (though,
for example, the word [r_ɔi] "sage" is transcribed as r^shi). Thus, it may
be useful as a base transliteration system.

The word "Malayalam" (pronounced in Malayalam as [m ay a ɻ a : j m])
itself presents a case study in the many transliterations that exist for the Ma-
layalam language. In English texts, of course, it is normally transcribed as
Malayalam; Maya and Varamozhi use the spelling malayaaLam; some-
times (particularly in pages in Finnish and some Eastern European langu-
ages), it may be transliterated as Malajalam (or some other word begin-
ning with Malajal-, e.g. Malajalә ina in Slovenian). Other translit-
erations, e.g. Alealum, Malayaalam, exist but may not be as common.
With other words for which there is no established spelling in any language
using the Roman script, the nature of possible transliterations is much more
complicated. For this reason, an additional method must be included in the
Malayalam version of Xiăozhī.

Generally, if Xiăozhī 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 Xiăozhī in any given
language. Nevertheless, the method can be applied to lesser studied lan-
guages with different transliteration information. Hopefully, we will inve-
sigate this aspect of transliteration more thoroughly on future occasions.

10 Conclusion

This paper describes a way to enhance web search for romanized trans-
literations of Mandarin words. This algorithm proposes a method that tran-
literates individual Mandarin Chinese words in a few different ways based
on three transliteration methods. The program first searches for an entire
term in a list of sets containing Pinyin sounds followed by the same sounds
in certain other transliteration systems. If the entire term is not included in
the list, the program subtracts one letter at a time from the end of the term
and searches the list again. After finding a part of the term that is included
in the list, the program repeats the process with the part of the term for which it has not yet found a corresponding set (or row). Finally, the program uses the rows and columns of the array to transliterate the term into other systems, searches, and keeps those combinations that are most common. This could help search engines improve their facilities for those searching for any information regarding a Chinese word, phrase, city name, etc. It is important to note that this algorithm is not limited, however, to Internet search engines; it could also be applied to search of research archives concerning lesser studied languages.
11 References


[Stalls and Knight] BG Stalls, K Knight, Translating Names and Technical Terms in Arabic Text, COLING/ACL Workshop on Computational Approaches to Semitic Languages, Montreal, Quebéc, 1998

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