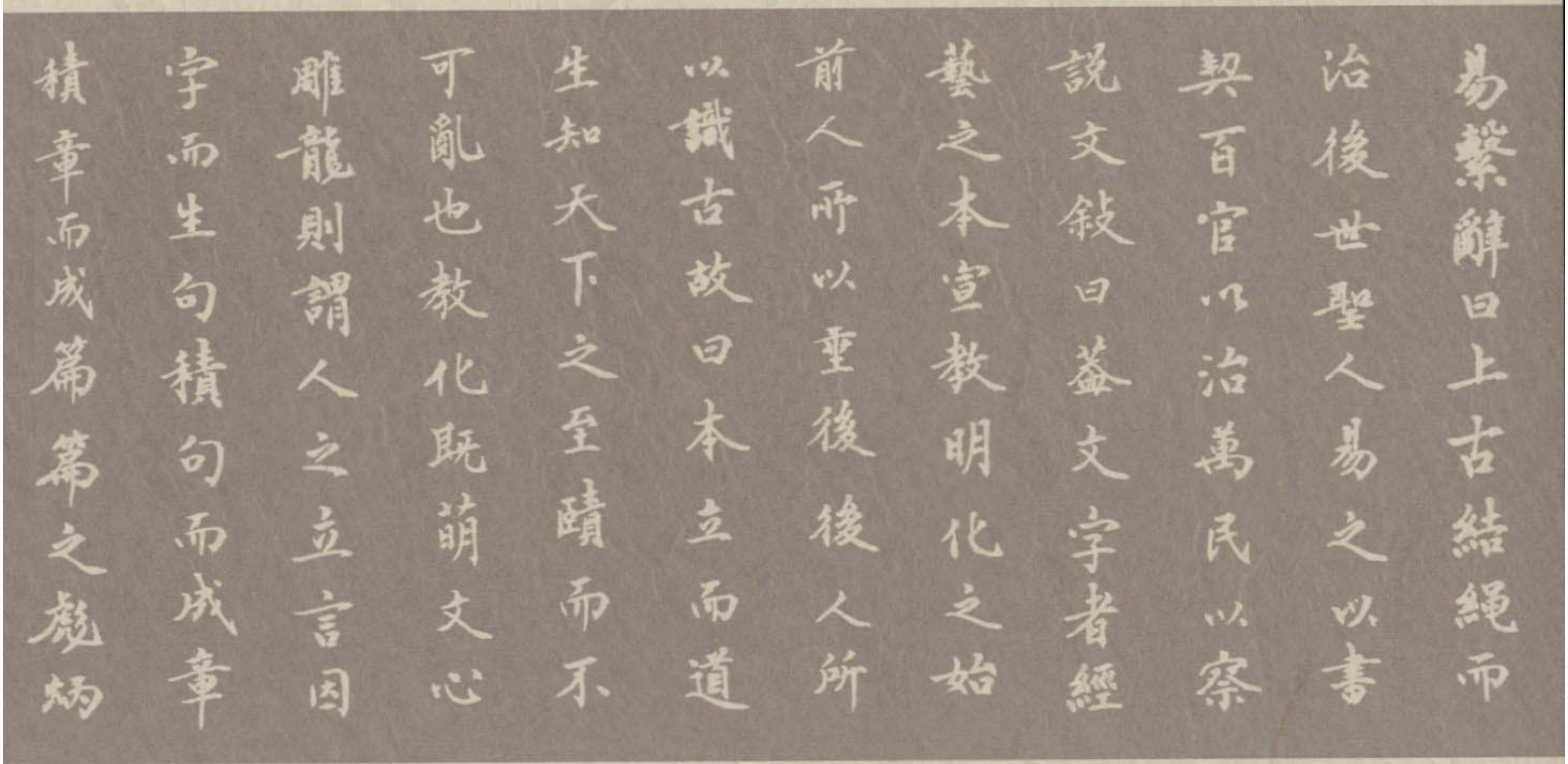
International Journal of

Computational Linguistics & Chinese Language Processing

中文計算語言學期刊

A Publication of the Association for Computational Linguistics and Chinese Language Processing

This journal is included in THCI Core, Linguistics Abstracts, and ACL Anthology.



Vol.15 No.2 June 2010 ISSN: 1027-376X

International Journal of Computational Linguistics & Chinese Language Processing

The Association for Computational Linguistics and Chinese Language Processing, Taipei

International Journal of

Computational Linguistics & Chinese Language Processing

The editing of this journal is subsidized by Center for Humanities Research National Science Council in 2010.

Aims and Scope

International Journal of Computational Linguistics and Chinese Language Processing (IJCLCLP) is an international journal published by the Association for Computational Linguistics and Chinese Language Processing (ACLCLP). This journal was founded in August 1996 and is published four issues per year since 2005. This journal covers all aspects related to computational linguistics and speech/text processing of all natural languages. Possible topics for manuscript submitted to the journal include, but are not limited to:

- Computational Linguistics
- Natural Language Processing
- Machine Translation
- Language Generation
- Language Learning
- Speech Analysis/Synthesis
- Speech Recognition/Understanding
- Spoken Dialog Systems
- Information Retrieval and Extraction
- Web Information Extraction/Mining
- Corpus Linguistics
- Multilingual/Cross-lingual Language Processing

Membership & Subscriptions

If you are interested in joining ACLCLP, please see appendix for further information.

Copyright

Cover

Calligraphy by Professor Ching-Chun Hsieh, founding president of ACLCLP
Text excerpted and compiled from ancient Chinese classics, dating back to 700 B.C.
This calligraphy honors the interaction and influence between text and language

International Journal of
Computational Linguistics and Chinese Language Processing
vol. 15, no. 2, June 2010

Contents

Papers

Computational Linguistics and Chinese Language Processing
Vol. 15, No. 2, June 2010, pp. 77-102

A Punjabi to Hindi Machine Transliteration System

Gurpreet Singh Josan*, and Gurpreet Singh Lehal*

Abstract

Transliteration is the general choice for handling named entities and out of vocabulary words in any MT application, particularly in machine translation. Transliteration (or forward transliteration) is the process of mapping source language phonemes or graphemes into target language approximations; the reverse process is called back transliteration. This paper presents a novel approach to improve Punjabi to Hindi transliteration by combining a basic character to character mapping approach with rule based and Soundex based enhancements. Experimental results show that our approach effectively improves the word accuracy rate and average Levenshtein distance of the various categories by a large margin.

Keywords: Transliteration, Punjabi, Hindi, Soundex Approach, Rule based Approach, Word Accuracy Rate.

1. Introduction

Every machine translation system has to deal with out-of-vocabulary words, like technical terms and proper names of person, places, objects, *etc*. *Machine transliteration* is an obvious choice for such words. When words cannot be found in translation resources, such as a bilingual dictionary, transliteration - the process of converting characters in one alphabet into another alphabet - is used. Transliteration is a process wherein an input string in some alphabet is converted to a string in another alphabet, usually based on the phonetics of the original word. If the target language contains all the phonemes used in the source language, the transliteration is straightforward, *e.g*. the Hindi transliteration of Punjabi word "ਕਮਰਾ" [kamarā] (room)[1] is "कमरा" [kamarā], which is essentially pronounced in the same way. Nevertheless, if some of the sounds are missing or are extra in the target language, they are generally mapped to the most phonetically similar letter, *e.g.*, in Hindi we have syllables (consonant clusters) that are written as a single atomic grapheme and have a double sound

* Department of Computer Science, Punjabi University Patiala, Punjab, India
Email: josangurpreet@rediffmail.com; gslehal@gmail.com
The author for correspondence is Gurpreet Singh Josan.

[1] The text in [ ] denotes phonetic transcription whereas in ( ) denotes the English translation of a word.

associated with them, like "क्ष", which is a combination of the sound of "क" and "ष". No such consonant cluster is present in Punjabi. So, a similar sounding letter generally is used to denote such sounds. Again, there is no rule to find out when a sequence of letters in Punjabi is going to map in a consonant cluster in Hindi, *e.g.,* consider the two names written in Punjabi, *viz*. ਸ਼ਿਤਿਜ [ṣitij] क्षितिज [kṣitija] and ਸ਼ਿਕਾਕਾਈ [śikākāī] शिकाकाई [śikākāī]. In the first name, the consonant cluster ਸ਼ਿ is mapped to क्षि, whereas, in the second name, the same is mapped to शि.

The important consideration in transliteration is to preserve the phonetics of transliterated word (Virga & Khudanpur, 2003). The transliteration process depends on the target language and on the education and experience of the actual transliterator. Thus, a single foreign word can have many different transliterations. For example, "ਮਹਿਫ਼ੂਜ਼" [mahiphūz] (safe) can be transliterated as "महफ़ूज़" [mahaphUz], "महफ़ूज़" [mahafUz], "महिफ़ूज़" [mahiphUz], "महिफ़ूज़" [mahifUz], *etc.* This variation, due to localization, poses problems in various NLP tasks, like cross language information retrieval. According to Viswanadha R. (2002), for a finer transliteration among two scripts, the scripts must have the features of completeness, predictability, pronounceable, unambiguousness, and partial reversibility. Transliteration schemes have to face the problem of letters present in one language and not in the other. Unless a superset of letters from all of the Indian Languages is formed, uniform transliteration is ruled out. Viswanadha, (2002) had the view that, when characters do not have any appropriate transliteration, they should be consumed and not replaced with any other character. This results in partial loss of reversibility and readability.

For several decades now, Roman transliteration has been used in English Indian language texts. Extensive research has been carried out on methodologies for transliterating Indian scripts to and from their Romanized counterpart. Transliteration among Indian scripts is a rather neglected area. With the development of ISCII (Indian Script Code for Information Interchange), the problems in transliteration among Indian scripts have been solved to some extent. ISCII has been designed via the phonetic property of Indian scripts and caters to the superset of all Indian scripts. The ISCII document is IS13194:1991, available from the office of the Bureau of Indian Standards. The ISCII approach is based on the fact that many Indian language groups share a common set of phonemes, and via the same set of codes for different Indian languages, it would be possible to change fonts from one language to the other, thereby "transliterating" the script in the process. By attaching an appropriate script rendering mechanism to ISCII, transliteration from one Indian script to another is achieved in a natural way. Later in the decade of the 1990s, Unicode standards were developed for representing the character set of all languages. In Unicode, alphabets are coded by script rather than language, which saves reduplication of the same character if it occurs in multiple languages. The Gurmukhi and Devanagari Unicode tables were based on templates from the 1988 versions of

the ISCII standard.

In this paper, we will discuss the Punjabi to Hindi Machine transliteration system. Although Punjabi and Hindi are related languages, and, except for a few cases, all letters of Gurmukhi script (a script for the Punjabi Language) are present in Devanagri script (a script for the Hindi Language), the task of transliteration from Punjabi to Hindi is not trivial. We will use letter to letter mapping as the **baseline** and try to find the improvements by rule based and Soundex based approaches. This work forms part of larger Punjabi to Hindi machine translation system, and the results of transliteration will be applied to this system for its improvement.

The remainder of the paper is organized as follows. Section 2 reviews previous work. In Section 3, the Gurmukhi and Devanagri scripts are compared, and the problems of transliteration from Punjabi to Hindi are discussed. Section 4 describes the formulation of the rule based model and the Soundex based model. Section 5 outlines the experiment and the evaluation methodology used. Section 6 will present the results. Finally, Section 7 contains our overall impressions and conclusions and points to future work.

2. Previous Work

The topic of machine transliteration has been studied extensively for several different language pairs, and many techniques have been proposed. Grapheme based models and Phoneme Based models are the two approaches found in literature. Grapheme refers to the basic unit of a written language: for example English has 26 graphemes or letters. Phonemes are the simplest significant unit of sound *e.g.* the /M/, /AE/, and /TH/ in *math.* Grapheme based models are classified into the statistical transliteration based model, decision tree based model, transliterated network based model *etc*. Some examples of this would be the noisy-channel model (NCM) (Lee & Chang, 2003; Virga *et al*., 2003), HMM (Jung, Hong & Paek, 2000), decision tree (Kang & Choi, 2000), transformation-based learning (Meng, Lo, Chen & Tan, 2001), statistical machine transliteration model (Lee *et al*. 2003), finite state transducers (Knight & Grahel, 1998), and rule-based approach (Oh & Choi, 2002; Wan & Verspoor, 1998). The phoneme-based approach has received remarkable attention in various works (Lee *et al*. 2003; Meng *et al*. 2001; Oh *et al*., 2002; Knight *et al*. 1998; Virga *et al*. 2003; Jung *et al*. 2000; Al-Onaizan & Knight, 2002).

For Indian languages, as mentioned earlier, Roman transliteration has been used to represent texts of Indian languages in English. Since it is difficult to represent the letters of Hindi using just the twenty six letters of the Roman alphabet, scholars have used varying schemes to accommodate sounds that could not be correctly indicated using appropriate Roman letters. The schemes are somewhat arbitrary in the choice of Roman letters. Sometimes, phonetic symbols are used in place of the normal Roman letters. Most phonetic symbols are

basically the letters of the Roman and Greek alphabets with special marks known as diacritic marks. Roman transliteration that makes use of diacritic marks works better for Indian languages, and some standardization has been effected in the last few decades based on the recommendation from the National Library in Calcutta ("Transliteration Principles," 2008). Roman letter assignments in this scheme are phonetically equivalent to the letters of Sanskrit or other Indian languages. The primary difficulty in data entry of the phonetic symbols is that there is no provision to input the symbols directly using the standard ASCII keyboard. Transliteration methods using only the displayable ASCII symbols do not run into this problem since the ASCII letters can be typed in directly. A special computer program, however, would be required to interpret the input string to produce the Indian languages display or printout. This is precisely what the currently popular transliteration schemes attempt. Schemes such as ITRANS (Chopde, 2001), RIT (Kanneganti & Kishore, 2008), ADHAWIN (Srinivasan, 1995), and MYLAI (KalyanaSundram, 2008), use only the standard displayable ASCII letters and symbols to transliterate the text. These schemes allow multiple representations for certain syllables and long vowels but the processing program handles this well. Although the input can be done using standard keyboards, these schemes are not suitable for searching the contents on the Internet. There is a need to evolve a case-insensitive transliteration scheme to facilitate searching on web.

For the Punjabi language, a Gurmukhi to Roman transliteration system using a transliteration scheme based on ISO: 15919 transliteration and ALA-LC has been developed at Punjabi University Patiala (Sharma, 2008).

All of these transliteration schemes are either from Roman to Indian languages or *vice-versa*. Transliteration among Indian languages is rather ignored. As the English language is one of the official languages of India, Roman transliteration is developed primarily to disseminate information in the English language. Little need is felt for transliteration among Indian languages. Malik (2006) has developed a machine transliteration system from Hindi to Urdu. The system converts Hindi and Urdu into a common language that may be ASCII encodings of Hindi and Urdu characters or the International Phonetic Alphabet (IPA) equivalents of Hindi and Urdu Phonemes. Then, Hindi and Urdu will be generated from the common language. This makes the system reversible.

A corpus based transliteration system for Shahmukhi to Gurmukhi has also been developed (Saini & Lehal, 2008). Their system uses statistical data from both Shahmukhi and Gurmukhi corpora like character, word, and n-gram frequencies. In this system, script mappings are performed for simple consonants, aspirated consonants (AC), vowels, and other diacritical marks or symbols. Next, the transliteration system is virtually divided into two phases. The first phase performs rule-based transliteration, and the second phase performs the transliteration using bi-gram language model.

A machine transliteration system has also been developed for transliterating from Hindi to Punjabi (Goyal & Lehal, 2009). This system has implemented approximately fifty complex rules for making the transliteration between Hindi-Punjabi language pair. The system claims 98% accuracy. The system is not reversible due to the dependency on language-specific rules.

As more and more data in Indian languages is available in electronic format, the need of the hour is to develop sophisticated transliteration modules for Indian languages that can be used in various NLP systems, like machine translation and cross language information retrieval. This paper discusses the first attempt to develop a transliteration system for transliterating Punjabi text to Hindi.

3. Gurmukhi and Devanagari Scripts

Punjabi can be written in Gurmukhi script or in Shahmukhi script. Shahmukhi is used to write and record the Punjabi language in West Punjab in Pakistan. In East Punjab in India, however, the Gurmukhi script is used to record the Punjabi language. This paper also considers this script for writing Punjabi. The script used for writing Hindi is called Devanagari. Both Gurmukhi and Devanagari descended from the Brahmi script of Ashoka. Both scripts are left-to-right and words are spelled phonetically. As a background for readers who are not familiar with script terminology, we will describe the basic elements of a script here. Building blocks of a script are alphabets, which are the sets of letters, each of which represents one or more phonemes in the language they are used to write. In some cases, combinations of letters are used to represent single phonemes, as in the English sh, ch, and th. The alphabets contain letters for both consonants (sound articulated with complete or partial closure of the vocal tract) and vowels (sound pronounced with an open vocal tract). A single vowel or a consonant plus a vowel make a syllable. A writing system can be syllabary or alphasyllabary. A syllabary system is a phonetic writing system consisting of symbols representing syllables. On the other hand, an alphasyllabary system consists of symbols for consonants and vowels. The alphasyllabary system is also known as abiguda. Both Gurmukhi and Devanagri are alphasyllabary in nature. Alphabets of both scripts represent syllables. All consonants contain an inherent vowel /a/ or schwa ending, both of which can be altered or muted by means of diacritics or *matra*. Vowels can also be written with separate letters when they occur at the beginning of a word or on their own. When two or more consonants occur together, special conjunct symbols are often used to add the essential parts of the first letter or letters in the sequence to the final letter (“Writing Systems,” 2010).

Gurmukhi, meaning "from the mouth of the Guru" is the most commonly used script in India for writing in Punjabi. Gurmukhi was introduced by the second Guru of the Sikhs, Guru Angad Dev Ji, in the sixteenth century as a polished version of the Landa writing system, which was used in old Punjabi at that time. Part of the shift from Landa to Gurmukhi entailed

the inclusion of some Devanagari letters. Gurmukhi was then used to record the scriptures of the Sikhs, and it continues till today as the prominent writing system for Punjabis. Bhatia (1993) identifies three stages of development the Punjabi language has undergone over the years, which are illustrated in Figure 1 below.

Stage 1 : Old Punjabi (10th to 16th century)
Stage 2 : Medieval Punjabi (16th to 19th century)
Stage 3 : Modern Punjabi (19th century to present)

Figure 1. The development stages of the Punjabi language.

In Gurmukhi, there are three letters that are used to provide bases for free-standing vowels:

ੳ [uṛa] ਅ [æṛa] ੲ [iṛi]

There are nine dependent vowel sounds (also called diacritics) available in Punjabi as ਾ, ਿ, ੀ, ੁ, ੂ, ੇ, ੈ, ੋ, and ੌ. The allowed combination of vowel symbols and sounds are ਅ, ਆ, ਇ, ਈ, ਉ, ਊ, ਏ, ਐ, ਓ, and ਔ. These are also called independent vowels. All of these combinations have a unique code in Unicode and are termed as independent vowels. When a vowel sound comes at the start of a word or is independent of some consonant in the middle or end of a word, independent vowels are used. There are 32 consonants in the older Gurmukhi script. Later, some more letters were introduced to represent some sounds adapted from Farsi and Persian. Presently, Gurmukhi has 38 consonants, 10 vowels letters (independent vowels), 9 vowel symbols (dependent vowels), 2 symbols for nasal sounds, and 1 symbol that duplicates the sound of consonants (Bhatia, 1993; Malik, 2006). The form of each syllable is often dependent on whether it occurs in combination with other letters, *e.g.*, a vowel following a consonant changes its form, behaving as a diacritic modifier to the consonant:

ਕ + ੀ = ਕੀ

All consonants have an inherent vowel, *i.e.* /a/. Diacritics, which can appear above, below, before or after the consonant they belong to, are used to change the inherent vowel. When certain consonants occur together, a special conjunct symbol *halant* character ੍ before the consonant is used to combine the essential parts of each letter. For example

ਕ + ੍ + ਰ = ਕ੍ਰ

These are also called conjunct consonants. A sentence illustrating Gurmukhi script is given below. Terminology used throughout the paper is: TT denotes phonetic transcription, G denotes gloss, and E denotes English sentence.

“ਪੰਜਾਬੀ ਮੇਰੀ ਮਾਂ ਬੋਲੀ ਹੈ।”

TT: pañjābī mērī māṃ bōlī hai.

G: Punjabi my mother tongue is.

E: Punjabi is my mother tongue.

The Nāgarī (lit. 'of the city') or Devanāgarī ('divine Nagari') was originally developed to write Sanskrit but was later adapted to write many other languages, including Hindi. Devanagari has 65 consonants, 18 full vowel letters, 17 vowel symbols, and 2 symbols for nasal sounds. Hindi uses only 34 consonantal syllables, 11 vowel letters (independent vowels), 9 vowel symbols (dependent vowels), and 2 symbols for nasal sounds (“The Devanagari Script Block,” 2008). Similar to Gurmukhi, the form of each syllable is often dependent on whether it occurs in combination with other letters, *e.g.*, a vowel following a consonant changes its form, behaving as a diacritic modifier to the consonant:

क [ka] + आ [ā] = का [kā]

When two consonants occur together, the combination results in a conjunct character. The first sign receives the diacritic ् known as a *halant*, to show that the consonant has its inherent vowel silenced, also known as the *dead consonant form*.

क् [ka] + र [ra] = क्र

Unlike Devanagari, there are only a few conjunct consonants in Gurmukhi. In Devanagari, if a consonant is followed by two vowels, then the first vowel is written in the diacritic form, whereas the second is written in full. This feature is not present in Gurmukhi. Vowels may also be nasalized, which is indicated by a anusvara “ ं” or *chandrabindu* “ ँ” written over the head stroke (horizontal line) of the vowel or the consonant it is attached to.

हुँ [hũ] nasalized

In cases where part of the vowel is written above the head stroke, only the dot is used to indicate nasalization. Although chandrabindu is not a part of Standard Hindi, in practice, both are used interchangeably in Hindi. The sentence illustrating Hindi is given below:

“पंजाबी मेरी मातृभाषा है।”

paṁjābī merī mātṛbhāṣā hai

G: Punjabi my mother tongue is.

E: Punjabi is my mother tongue.

Except for minor differences, most of the letters are same in both of the scripts. There are three syllables of consonant clusters in Hindi that are written as a single atomic grapheme, *i.e.*, त्र , ज्ञ, क्ष but no such letters or consonant clusters are available in Gurmukhi. Punctuation in

Punjabi is similar to Hindi. Table 1, adapted from (Goyal & Lehal, 2009), shows the similarities and dissimilarities among alphabets in both scripts. As we can see, most of the letters have one to one correspondence, so this table also forms the base for direct mapping.

Table 1. Alphabet of Gurmukhi and Devanagari scripts

Gurmukhi	Devanagari	Gurmukhi	Devanagari	Gurmukhi	Devanagari	Gurmukhi	Devanagari	Gurmukhi	Devanagari	Gurmukhi	Devanagari	Gurmukhi	Devanagari
ੳ	-	◌ਾ	◌ा	ਕ	क	ਟ	ट	ਨ਼	ऩ	ਲ	ल	ਗ਼	ग़
ਅ	अ	◌ਿ	◌ि	ਖ	ख	ਠ	ठ	ਪ	प	ਲ਼	ळ	ਜ਼	ज़
ੲ	-	◌ੀ	◌ी	ਗ	ग	ਡ	ड	ਫ਼	फ़	-	ऴ	ੜ	ड़
ਆ	आ	◌ੁ	◌ु	ਘ	घ	ਢ	ढ	ਬ	ब	ਵ	व	ਫ	ढ़
ਇ	इ	◌ੂ	◌ू	ਙ	ङ	ਣ	ण	ਭ	भ	ਸ਼	श	ਭ	फ़
ਈ	ई	◌े	◌े	ਚ	च	ਤ	त	ਮ	म	-	ष	ਯ਼	य़
ਉ	उ	◌ੈ	◌ै	ਛ	छ	ਥ	थ	ਯ	य	ਸ	स	ੜ	ऱ
ਊ	ऊ	◌ੋ	◌ो	ਜ	ज	ਦ	द	ਰ	र	ਹ	ह	-	ऩ
ਏ	ए	◌ੌ	◌ौ	ਝ	झ	ਧ	ध	ਰ਼	ऱ	ਕ਼	क़	ਹ	ह़
ਐ	ऐ	-	◌ॄ	ਞ	ञ	ਨ	न	-	ऱ	ਖ਼	ख़	ਵ	व़
ਓ	ओ	-	◌ॢ	-	◌ृ								
ਔ	औ	-	◌ॣ	ੰ	-								

4. Approach to Transliteration from Gurmukhi to Devanagari

4.1 Letter to Letter Mapping

Both Punjabi and Hindi are phonetic languages, and their scripts represent the phonetic repository of their respective languages. These phonetic sounds are used to determine the relations between the alphabets of the two scripts. On the basis of this idea, character mappings are determined. The development of the Unicode system for representing alphabets eases the problem to some extent. With this system, every letter can be uniquely mapped to the corresponding letter, as shown in Table 1. Taking into account the similarity of both of the scripts, letter to letter mapping is the obvious choice for the baseline computation. Letters are mapped using Table 1. This system is used as the baseline for improvements by the rule based

and Soundex based approaches.

For analysis and comparison purposes, both scripts are subdivided into different groups on the basis of types of letters, *e.g.*, consonants, vowel symbols, vowel sounds, and other symbols.

4.1.1 Vowel Mapping

Punjabi contains 10 vowel symbols. Hindi vowels have one to one correspondence with Punjabi vowel symbols. There are nine dependent vowel sounds available in Punjabi as ਾ, ਿ, ੀ, ੁ, ੂ, ੇ, ੈ, ੋ, and ੌ. Corresponding vowel sounds in Hindi are ा, ि, ी, ु, ू, े, ै, ो, and ौ. When a vowel sound comes at the start of a word or is independent of any consonant in the middle or the end of the word, independent vowels are used. The mapping is shown in Table 2.

Table 2. Vowel Mapping

Independent vowels		Dependent vowels	
Gurmukhi	Devanagari	Gurmukhi	Devanagari
ਅ	अ	ਾ	ा,
ਆ	आ	ਿ	ि,
ਇ	इ	ੀ	ी,
ਈ	ई	ੁ	ु,
ਉ	उ	ੂ	ू,
ਊ	ऊ	ੇ	े,
ਏ	ए	ੈ	ै,
ਐ	ऐ	ੋ	ो,
ਓ	ओ	ੌ	ौ
ਔ	औ		

Besides these vowels, Punjabi has two more symbols, ੳ and ੲ. No symbol is present in Hindi for these two symbols. The nearest phonetic letters are उ and इ, respectively. These are used in the letter to letter mapping scheme.

4.1.2 Consonant Mapping

Consonant mapping is shown in Table 1, Columns 5-14. No letter in Punjabi is present for the Hindi letters "क़", "ष", and "स़". This means these letters can never be mapped in a letter to letter based approach. Similar is the case for some double-sound-producing syllables, like "क्ष", "ज्ञ", and "श्र". Syllables like "ज्ञ" and "श्र" can be mapped by a rule based approach while others like "ष", "क्ष", "क़", and "स़" can be handled by the Soundex based approach discussed in the next sections.

4.1.3 Conjunct Consonants Mapping

There are three conjunct consonants in Gurmukhi also called PAIREEN, ੍ਹ ("*Haahaa*"), ੍ਰ ("*Raaraa"*), and ੍ਵ ("*Vaavaa*"), which are shown in Table 3.

Table 3. Sub Joins of Gurmukhi

Conjunct Consonants	Punjabi	Hindi	English
੍ਹ	ਬੁਲ੍ਹੜ [bulharh]	बुल्हड़	A person with large lips
੍ਰ	ਪ੍ਰੀਤਮ [pritam]	प्रीतम	A person name (Pritam)
੍ਵ	ਸ੍ਵਰਗ [svarag]	स्वरग	Heaven

The usage of PAIREEN *"Haahaa"* is quite frequent in Punjabi, but the other two are very rare in usage and are used in Sanskrit loan words. In Punjabi, they are represented by the *halant* character ੍ before the consonant, which indicates that the inherent vowel is omitted. A similar *halant* character is also present in Hindi, which can replace the *halant* character in Punjabi. It may be noted that the actual sequence of letters is the same in both languages but the visual sequence of letters differs (Table 4). PAIREEN *"Haahaa"* in Punjabi is replaced with their full consonant counterparts, while the full consonant preceding them in Punjabi is shown in half form in Hindi. Similar is the case with PAIREEN "Vaavaa". On the other hand, PAIREEN "*Raaraa"* takes the position below the previous consonant in Punjabi as well as in Hindi.

Table 4. Actual Sequence of letters

ਬ	ੁ	ਲ	੍	ਹ	ੜ	=	ਬੁਲ੍ਹੜ
ब	ु	ल	्	ह	ड़	=	बुल्हड़

4.1.4 Other Symbols

Punctuation marks and digits are the same in both scripts. A special character called as visarga (ः) is present in Hindi but not in Punjabi. So, it will never be mapped in the letter to letter

based scheme. Besides this, Gurmukhi has two separate nasal characters, Bindi (ਂ) and Tippi (ਁ). Hindi has only one nasal character called anusvara " ं". In the Devanagari script, anusvara is represented with a dot (*bindu*) above the letter *e.g.* "मं". Both nasal characters of Punjabi are mapped to this single nasal character in Hindi. Adhak (ੱ) is used to duplicate the sound of a consonant in Punjabi. No such character is present in Hindi. Sound duplication is represented by half form consonants in Hindi.

4.1.5 Problems in Letter to Letter Mapping

The total number of letters in Punjabi and Hindi are not same (Table 1). The Punjabi letters ੳ and ੲ have no mapping in Hindi. Similarly, there are letters in Hindi that have no mapping in Punjabi *e.g.* ऋ. These letters will never be mapped in Punjabi to Hindi transliteration using a direct mapping method. Some letters have more than one representation in Hindi, *e.g.,* श may be mapped to श or ष. Another problem is the use of conjunct consonant forms in Hindi. In Hindi, a syllable may consist of a vowel, a consonant followed by vowel, or a consonant cluster followed by a vowel. The last form *i.e.,* when two or more consonants are used within a word with no intervening vowel sound, is known as a conjunct consonant. Use of conjunct consonants is limited in Punjabi. Only three letters can be used as conjuncts *i.e.*, ਹ, ਰ, and ਵ. Their representation is also unique. It is not a trivial task to find out which combinations of letters in Punjabi will take conjunct consonant form in Hindi. For example, why the word ਨਿਊ [niū] (new) in Punjabi takes the conjunct consonant form in Hindi न्यू, is not clear.

Also, the mapping of nasal consonants is not clear. Nasal consonants in initial place in a conjunct may be expressed using the anusvara over the previous vowel, rather than as a half-glyph attached to the following consonant. The anusvara is written above the headstroke, at the right-hand end of the preceding character. In the list below, both spellings are correct and equivalent, although anusvara is preferred in the case of the first two: रंग = रङ्ग, पंजाबी = पञ्जाबी, हिंदी = हिन्दी, लंबा = लम्बा. Anusvara is still applied when previous character has its own vowel sign. If the vowel sign is [aa], the anusvara appears over the [aa], *e.g.* फ़ांसीसी or आंदोलन.

Similarly, the position of the character "र" is not specific. It can be used as consonant, subscripted consonant, or can take a position above a consonant or diacritics, and it is typically displayed as a small mark above the *right* shoulder of the last letter in the syllable. It is not clear how the character ਰ [ra] in Punjabi will map to which form in Hindi, *e.g.*, the three cases where mapping of ਰ is not clear are: from ਤਰਕਸ਼ [tarkash] (arrow-holder) to तरकश, from ਵਰਤ [varat] (fast) to व्रत, and from ਬਰਤਨ[bartan] (utensil) to बर्तन.

4.2 Rule Based Approach

Character mapping alone is not sufficient for a Punjabi to Hindi machine transliteration system. Quite a reasonable improvement can be achieved by small amount of dependency or contextual rules. These rules are manually crafted with the help of a linguist by observing the problems in the direct mapping system. This section discusses such rules for alleviating some of the problems discussed in the previous section. Ideally, the rule set must contain all of the rules required by the system to perform, but practically it is not possible to construct such a set. About 11 rules are identified that are applicable on the source text and about 8 rules are identified that are applicable on the target text. We tried to cover all the possible cases but still we may miss some of them. Moreover, new rules can be added when identified. The rules go as follows:

1. Letters whose mapping is not available in Hindi :
   a. ੳ and ੲ are two such letters whose corresponding Hindi letters are not available. They are replaced by their most phonetically equivalent letters, *i.e.*, उ and इ respectively. Formally, the rule goes like this

      if inpt_str contain "ੳ" then replace it with "उ"

      if inpt_str contain "ੲ" then replace it with "इ"

   b. A character adhak "ੱ" is present in Punjabi and used to show the stress on the next character. No letter in Hindi is present to represent this character. Nevertheless, the purpose is served by placing a half character before the stressed character, *e.g.,* "ਸ਼ੱਕਰ" [śakkar] (jaggery) is transliterated as "शक्कर"[śakkar]. There is an exception for this rule. If the next character of adhak "ੱ" is ਖ [kha] then, instead of placing a half character, a half क[ka] is placed, *e.g.*, "ਮੱਖਣ" [makkhaṇ] (butter) transliterated to "मक्खन" [makkhana]. Also, if the next character is ਛ [ccha], then the half character is replaced by a half च, *e.g.*, as in "ਮੱਛਰ" [macchar] (mosquito), which is transliterated to "मच्छर" [macchar]. Similarly, if the next character is ਧ [dha], then the half character is replaced by a half द, *e.g.,* as "ਬੱਧ" [baddh] is transliterated as बद्ध. Formally, it is

      If current_char = "ੱ" then
        If nxt_char= "ਖ" then
            map "ੱ" to "क्"
        else if nxt_char= "ਛ" then
            map "ੱ" to "च्"
        else if nxt_char= "ਧ" then
            map "ੱ" to "द्"
        else
            map "ੱ" to nxt_char & "्"
        end if
      end if

2. Bindi and Tippi both are used to produce a nasalized sound in Punjabi. There is only Bindi (anusvara) present in Hindi to serve the same purpose. Tippi can be successfully mapped to Bindi in Hindi. A special case is when the tippi " ◌ੰ" [ṃ] is followed by "ਨ" [na] or "ਦ" [da], then it is replaced by a half "न", as in "ਕੰਨੜ" [kannaṛ] (Kannad) "कन्नड़" [kannaṛ]. If, however, it is followed by "ਬ" [ba], then it is replaced by a half "म", as in "ਮੋਜ਼ੰਬਿਕ" [mōzmbik] (mozambique) "मोज़म्बिक". Formally, it is

```
If current_char = "◌ੰ" then
  If nxt_char= "ਨ" then
        map "◌ੰ" to "न्"
  else if nxt_char= "ਬ" then
        map "◌ੰ" to "म्"
  else
        map "◌ੰ" to "◌ं"
  end if
end if
```

3. The presence or absence of the diacrtic mark "ਿ" in the target string is effected by the character ਹ [ha] in Punjabi. When ਹ [ha] in the input string is followed or preceded by the ਿ [i] mark, ਿ [i] is translitrated differently. In this case, ਹ [ha] is followed by ਿ then ਿ is ommited from the transliterated text, *e.g.*, ਸ਼ਹਿਰ [shahir] (city) is transliterated to शहर. On the other hand, if ਹ [ha] is preceded by ਿ, then ਿ is mapped to े in the transliterated text, *e.g.*, ਸਿਹਤ [sihat] (health) is transliterated to सेहत. Formally, it is

```
If current_char = "ਹ " then
  If nxt_char= "ਿ" then
        Omit "ਿ" in target string
  else if prev_char= "ਿ" then
        map "ਿ"  to "े"
  else
        map "ਿ" to "ਿ"
  end if
end if
```

4. Another rule deals with transliteration of letters ਅਰ [ar]. If this combination appears at the final position in a word, then, instead of mapping ਅ [a]to अ, this letter is mapped to य., *e.g.*, ਬੀਅਰ [bīar] (bear) is transliterated to बीयर. Formally, it is

If last_two_char = "ਅਰ " then

map "ਅ" to "य"

else

map "ਅ" to "अ"

end if

5. If ਅ is in the last position, then it is replaced by य as in ਪ੍ਰਿਅ [pria] – प्रिय Formally, it is

   If last_char = "ਅ " then map "ਅ" to "य"

6. If ਈ is in the last position, then it is replaced by यी, *e.g.*, in ਵਾਜਪਾਈ [vājapāī] (Vajpai)– वाजपायी Formally, it is

   If last_char = "ਈ " then map "ਈ" to "यी"

7. If ਓ is in the last position or the 2nd to last position, then it is replaced by यो, *e.g.*, in ਗਲੀਲਿਓ [galīliō] (galilio) – गलीलियो. Formally, it is

   If last_char = "ਓ" or second_last_char= "ਓ" then map "ਓ" to "यो"

8. A rule for plural forms of Punjabi words calls for changing the vowel sign of the last chararcter. It says that, if the last characters are ◌ਾ ◌ੰ, then ◌ਾ is mapped to ◌ੋ in Hindi, *e.g.*, ਕਿਤਾਬਾਂ [kitābāṃ] (books) is transliterated to किताबों[kitābõ]. Although this is a case of translation where the root ਕਿਤਾਬ [kitāb] is the same in the two languages and gets inflected by the rules in respective languages, this case can be handled at the transliteration stage. Formally, it is

   If last_two_char = "◌ਾ ◌ੰ" then map "◌ਾ" to "◌ो"

9. Similarly, if the last positions are occupied by ਆਂ [āṁ], then ਅ [a] is mapped to य, *e.g.*, ਸ਼ੀਸ਼ੀਆਂ [shīshīāṃ] (bottles) should be mapped to शीशीयों rather than शीशीओं. It is worth mentioning here that both of the spelling variants are acceptable in general to Hindi speakers but the former is preferred. Formally, it is:

   If last_two_char = "ਆਂ" then map "ਅ" to "य"

10. The character ੴ [ik-ōṅkār] is replaced by string "एक ओंकार". Formally,

    If char = "ੴ" then map "ੴ" to "एक ओंकार"

11. The letter "ਣ" [ṇ] is converted to "न" if it is a last consonant in a word, *e.g.*, in ਕਹਿਣਾ [kahiṇā] (to say) the letter ਣ is converted to न when transliterated to produce "कहना".

    If last_consnt = "ਣ" then map "ਣ" to "न"

The above rules are applicable for the character sequence found in the source script. Table 5 describes some rules that are applicable to the character sequence found in the target script, *i.e.* Hindi. We can find a probable character sequence using the letter to letter mapping approach. These rules then can be applied to these probable strings. Every row of Table 5 gives the

replacement rules for a character combination found in the probable string. For example, for a source script word ਵਿਉ [viu] (view), the letter to letter mapping approach produces a probable string विउ, which is converted to व्यु by applying the rule from the following table. Formally, it can be described as follows:

For each chr_seqnce in table

If chr_ seqnce present in input string then

Replace chr_ seqnce to corresponding seqnce from table

Table 5. Replacements in transliterated text

Occurrence in hindi token	Replacement in hindi token	Example
"ਿਉ"	"੍ਯੁ"	ਵਿਉ [viu]–विउ – व्यु
"ਿਊ"	"੍ਯੂ"	ਨਿਊ [niū]– निऊ – न्यू
"ਿਓ"	"੍ਯੋ"	ਬਿਓਰਾ [biōrā]– बिओरा – ब्योरा
"ੀਆ"	"ਿਯਾ"	ਕੋਰੀਆ [kōrīā]– कोरीआ – कोरिया
"ੀਅ"	"ਿਯ"	ਕੋਰੀਅਨ [kōrīan] — कोरीअन — कोरियन
"�इਣ"	"ਯਣ"	ਰਾਮਾਇਣ [rāmāiṇ] — रामाइण — रामायण
"ੀਯੋ"	"੍ਯੋ"	ਟੋਕੀਓ [ṭōkīō] — टोकीयो – टोक्यो
"ਿਆ"	"੍ਯਾ"	ਸਿਆਚਿਨ[siācin]— सिआचिन — स्याचिन

4.3 Soundex Based Approach

Considerable improvements are noticed over the letter to letter mapping approach when using the rule based approach, but still some problems persist. Although we tried to cover the maximum number of rules, still there are cases for which no rule applies. Letters or syllables not available in Punjabi but present in Hindi, ऋ, क्ष ,ज्ञ ,श्र ,ष, *etc*. are still unhandled. The syllables having double tones in Hindi, like ऋ is a combination of sound of consonant "र" /r/ and vowel "ि" /i/, are represented in Punjabi with the help of two letters, like the consonant "ਰ" /r/ and vowel "ਿ"/i/. There is no means to find when these two letters in Punjabi will map to corresponding letters "रि" or to a single syllable "ऋ" in Hindi, *e.g.,* consider following cases.

Case I: ਰਿਕਸ਼ਾ(riksha) [rickshaw] → रिक्शा

Case II: ਰਿਸ਼ੀ(rishi) [cleric] → ऋषि

In Case I, the letters ਰਿ are mapped to रि, whereas, in Case II, they are mapped to ऋ. Also, the rule based approach is deficient in producing the half form of letters and some other letters, as discussed in the previous section. For generating these characters, the Soundex technique is employed. Soundex is a phonetic matching technique. The Soundex algorithm was designed by Odell and Russell (1918) to find spelling variation of names. It represents classes of sounds that can be lumped together. Two names match if they have the same Soundex representation.

Back in 1918, Odell and Russell proposed Soundex, the first phonetic encoding for English to be used in the US census. Soundex partitions the set of letters into seven disjoint sets, assuming that the letters in the same set have similar sounds. Each of these sets is given a unique key, except for the set containing the vowels and the letters h, w, and y, which are considered to be silent and are not taken into account during encoding. For example, both *realize* and *realise* have been encoded to 'R-420' in Soundex encoding, which works well for trivial cases, but fails to give same code to words where letters change its pronunciation in different contexts. For example, *knight*, *night,* and *nite* are similar sounding words but Soundex does not give the same code to these words.

For Punjabi to Hindi transliteration, the Soundex concept is extended for searching for the correct spelling variant of a given transliteration. The Soundex codes are assigned to the Hindi characters based upon their phonetics *i.e.,* similar sounding characters get the same code, *e.g.,* both श and ष have the same sound, thus, they get the same code, 44. The akharas having single graphemes get the code according to the sounds they produce, *e.g.,* ऋ is a combination of sound of consonant "र" /r/ and vowel "ि" /i/, so it gets the code of these two letters, *i.e.* 4149. The code table is shown in Table 6. Codes in this table are selected in such a way that similar sounding letters get the same code. We start with independent vowel sounds whose code ranges from 1 to 5. For consonants, we start from two digit codes, *i.e.* 11 to 47. Thereafter, we code the dependent vowels and some other miscellaneous letters. Unlike Soundex code, which consists of a letter (the first letter of the string) followed by three numerical digits encoding the remaining consonants, all characters of the Hindi string are coded into digits. This ensures matching in case the first letter does not match. From the transliterations produced by the previously discussed methods, the correct spelling variant is searched using Soundex codes as described in the following lines. Let *H* be a set of equivalent Hindi words and *TH* be a transliteration produced by the machine transliteration step. A relevant *h* from *H* is selected by comparing phonetic similarity between *H* and *TH*. For implementation purposes, we make a unigram table from a corpus of about 1 GB size obtained from the Internet. Unigram contains about 150,000 unique words along with their frequency in the corpus. The codes are generated for each of the word in unigram using Table 6. For the comparison, letters in *TH* are converted into phonetic code using the mapping table as described in Table 6. Then, this code is looked for in a unigram table. The unigram table may

produce more than one candidate. The candidate with maximum frequency is selected as the correct variant of the given *TH* produced by the machine. For example, consider the word "ਡਰਾਫਟ" [ḍarāphaṭ] (draft) written in Punjabi. The string "डराफट" is produced by the baseline module. The code 2541483623 is generated for this string, as shown in the following table.

ड	र	ा	फ	ट
25	41	48	36	23

This code is looked for in a unigram database. This database contains two entries against this code, *i.e.* ड्राफ्ट and डराफ्ट with frequency 12 and 8. The string with higher frequency is selected as the correct output for this input.

Table 6. The coding scheme for Hindi text

अ	1	च	17	न	34	ा	48
ऑ	1	छ	18	प	35	ॉ	48
इ	2	ज	19	फ	36	ि	49
ई	2	ज़	19	फ़	36	ी	49
उ	3	ज्ञ	20	ब	37	ो	50
ऊ	3	झ	21	भ	38	ौ	50
ए	4	ञ	22	म	39	ू	51
ऐ	4	ट	23	य	40	ु	51
ओ	5	ठ	24	र	41	े	52
औ	5	ड	25	ऱ	42	ै	52
क	11	ड़	26	ल	42	ँ or ं	53
क़	11	ढ	27	ळ	42	़	54
क्ष	12	ढ़	28	व	43	ॐ	55
ख	13	ण	29	श	44	ॠ	4149
ख़	13	त	30	ष	44	ॠ	4149
ग	14	थ	31	आ	45	ृ or ॄ	4149
ग़	14	द	32	स	46	श्र	4441
घ	15	ध	33	ह	47		
ङ	16						

5. The Experiment

In this section, we describe the evaluation of our models on the task of Punjabi to Hindi transliteration.

5.1 Data

A provocative question for evaluators is the type of test material that should be adopted for evaluation. Balkan (1994) describes three types of test material, *viz*. test corpora, test suites, and test collections. While test corpora is a collection of naturally occurring texts, a test suite is a collection of artificially constructed inputs, where each input is designed to probe a system's treatment of a specific phenomenon or set of phenomena. A test collection is a set of inputs associated with a corresponding set of expected outputs. Test suites and test corpora are extensively used for evaluating various systems involving NLP, while test collections are rarely used.

Users can make their own test suites, and such test suites are helpful for checking the improvement of an MT system. For two competing systems, however, test suites are not considered a good method of assessment. It is difficult to interpret the result that both systems transliterate same percentage of text but fail on different sentences. How to design a test suite, the cost of constructing it, and what sort of sentences/words should go into a test suite are some other issues discussed by Arnold (1995). There are several issues involved in the selection of a set of words for a comprehensive evaluation. For example, the set could be constant, variable, or a mixed one; the collection of words may be domain-specific or generic. It is obvious that there is no guarantee that even the bulkiest sample will include all of the possible words of the source language. Some suggestions as to what should go into a test suite are discussed in Hoard (1991). Some empirical evidence is present in literature to answer the question of how much text to include in a test suite. Although this evidence is not for transliteration system, it can provide insight for this system as well. Elliott *et al*. (2003) tried to prove the general hypothesis that more text would lead to more reliable scores. Based on an empirical assessment of score variation, the authors estimate that systems could be reliably ranked with test suite of around 14,000 words. Zhang & Vogel (2004) also studied the influence of the amount of test data on the reliability of automatic metrics, focusing on confidence intervals for BLEU and NIST scores. Their results show that BLEU and NIST scores become stable when using around 40% of the data (around 40 documents or 300 sentences), although stability is defined here in terms of the distance between scores of different systems. Estrella *et al*. (2007) shows that, for human or automatic evaluation, about five documents from the same domain-with 6,000 words-seem sufficient to establish the ranking of the systems and about ten documents are sufficient to obtain reliable scores.

For this experiment, a benchmark sampling method is used to select the set of words. Unicode encoded text is used in Punjabi as well as Hindi, as it eliminates the need of normalizing the text. If text is in font encoded form, then the ASCII code for a given character may vary. For example, under the AnmolLipi font, the Latin character 'A' would appear as 'ਅ'. Conversely, under the DrChatrikWeb font, it would appear as 'ੳ'. Unicode encoded text provides a unique way of representing a character; hence, it eases the task of mapping letters from different alphabets. We tested on people names, location names, and foreign words in Punjabi. The list was prepared manually by typing the names and terms from telephone directories, census records, *etc*. and partially obtained from manually searching the Internet. We tried to include the maximum possible variety of words in the set. The words are further categorized as Punjabi names, Hindi names, English names, and other foreign names/words. It is worth mentioning here that quite a number of English words are adapted into the Punjabi and Hindi languages, so we categorized them separately from the foreign words, which are words from other languages. The list contains about 3500 person names, 1500 location names, and 1000 foreign words.

5.2 Evaluation Methodology

The following combinations of approaches are tested for Punjabi-Hindi transliteration task.

Baseline: As a baseline for our experiments, we used a simple letter to letter based approach that maps Punjabi letters to the most likely letter in Hindi. We call it CASE-I.

Baseline followed by Rule Based Approach: Some hand crafted rules are studied for the improvements in the letter to letter based mapping system. This system is called CASE-II.

Baseline followed by Soundex Approach: The Soundex approach is applied to the output of the baseline method. This is termed as CASE-III.

Baseline followed by Rule Based followed by Soundex Based Approach: Both the rule based and Soundex approaches, consecutively, are applied to the output of the baseline method. This is termed as CASE-IV.

Baseline followed by Soundex Based followed by Rule Based Approach: Both the Soundex based and Rule based approach, consecutively, are applied to the output of the baseline method. This is termed as CASE-V.

Human: For the purpose of comparison, we allowed an independent human subject (fluent in Punjabi and native speaker of Hindi) to perform the same task. The subject was asked to transliterate the Punjabi words in the test set without any additional context. No additional resources or collaboration were allowed. This output was used as the gold standard for checking the system's performance.

5.3 Evaluation Metrics

Often, more than one transliteration is acceptable for any input string. Using one gold standard may prove to be too rigid for measuring accuracy. On the other hand, including more than one correct transliteration complicates the computation of the evaluation score. Also, there is no one exact transliteration of a foreign word, so a soft standard is required to indicate the closeness of output to the gold standard. Levenshtein distance is used for this purpose. The metrics are described as follows:

Word Accuracy Rate: It is defined as percentage of correct transliteration from the total generated transliterations by the system.

Average Levenshtein Distance: This is the average of Levenshtein distances between the transliterated word and reference word. Levenshtein distance is a measure of similarity of two strings. The distance is the number of deletions, insertions, or substitutions required to transform the source string to the target string, *e.g.,* transforming from the source string इखतिआर, which results from the direct transliteration of ਇਖਤਿਆਰ [ikhtiār], the correct target string इख्त्यार can be produced by deleting ि from त, adding halant (्) to त, and replacing अ with य. Thus, the Levenshtein distance is 3.

While Levenshtein distance captures the performance of a system at character level, word accuracy rate is used to capture the performance at word level. A word may have non-zero Levenshtein distance but still may be accurate. Lower Levenshtein distance value implies that, although the result is not the same as that of the gold standard, it is still intelligible.

6. Evaluation Results

The experiment was performed on the prepared list. The results of this experiment follow.

Table 7. Word Accuracy Rate and Avg Lev Dist. of Fivr Cases

	CASE-I	CASE-II	CASE-III	CASE-IV	CASE-V
WAR	73.13%	78.88%	86.72%	92.65%	87.15%
ALD	0.61	0.32	0.39	0.10	0.37

The baseline model produced a 73.13% accuracy rate. The rule based enhancement and Soundex based enhancement shows the improvements in performance with Soundex prove to be better than rule based. It is observed that CASE-V, *i.e.* Soundex followed by rule based, does not provide much improvement. This is because in the literature, the Soundex approach is largely discussed as a spelling correction method. Thus, it performs better on refined text, *i.e.* the output generated by the rule base, and fails to perform on raw input. Similarly, the average edit distance of different cases is shown in Table 7. The average edit distance of

CASE-III is increased from CASE-II then decreased with WAR increasing, as evident from Figure 2. This is because the human evaluator has marked some spelling variations correct. That is, for the name ਸੁਰਿੰਦਰ (*Surinder*), the three transliterations produced by CASE-I, Case-II, and Case-III are सुरिंदर, सुरिन्दर and सुरिंद्र, respectively. The human evaluator marked all of these as correct. Nevertheless, as Levenshtein distance is measured against a gold standard, which is सुरिंद्र in this case, CASE-III shows an increase in this parameter despite increased word accuracy. The word accuracy rate for CASE IV, *i.e.* the Base + rule + Soundex approach is found out to be 92.65% and Avg. Levs. Dist is 0.10. These figures are the highest among all the cases and verify the suitability of the approach.

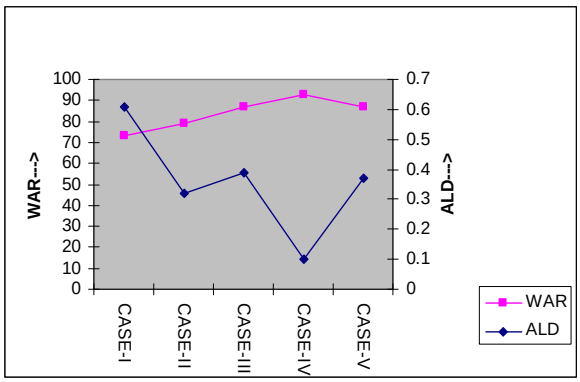


Figure 2. Word Accuracy Rate and Avg Lev Dist. of Five Cases

The breakdown of the figures is shown in following tables.

Table 8. WAR and ALD for person names, location names, and foreign words in all cases

	Person Name		Location Name		Foreign Words	
	WAR (%)	ALD	WAR (%)	ALD	WAR (%)	ALD
CASE-I	75.85	0.59	67.10	0.61	63.50	0.64
CASE-II	86.9	0.23	79.8	0.24	69.8	0.50
CASE-III	88.5	0.41	81.7	0.51	89.8	0.26
CASE-IV	92.8	0.1	91.45	0.1	93.8	0.09
CASE-V	88.78	0.39	82.7	0.48	90.01	0.24

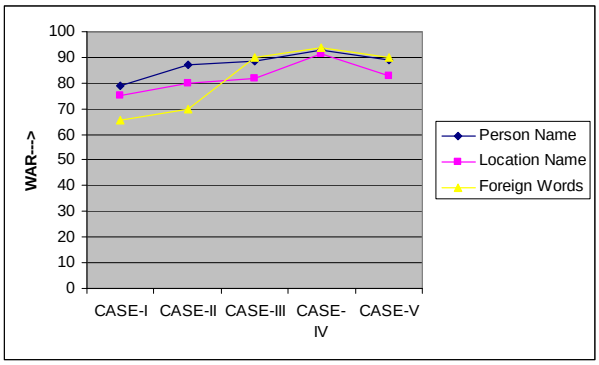


Figure 3. WAR for person names, location names, and foreign words in all cases

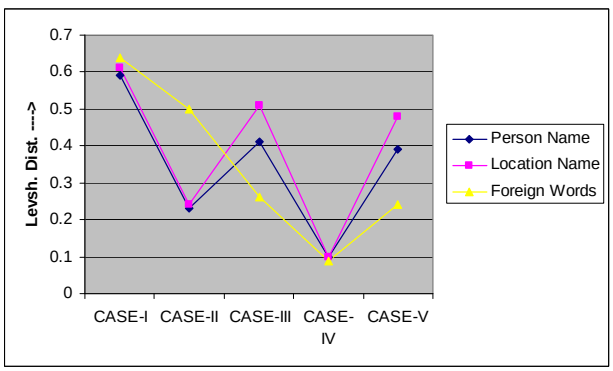


Figure 4. ALD for person names, location names, and foreign words in all cases

The word accuracy for words whose origin is also Punjabi is 86.6 in the baseline model and shows the similar trend in other cases, as shown in Table 9. The improvement in all cases is registered with maximum for Hindi and English languages.

Table 9. Word Accuracy rate and Avg. Lev Dist according to origin of source language of input word

	Punjabi Word		Hindi Word		English Word		Foreign Word	
	WAR	ALD	WAR	ALD	WAR	ALD	WAR	ALD
CASE-I	86.6	0.41	63.54	0.76	57.3	0.83	80.1	0.39
CASE-II	91.6	0.12	70.15	0.44	63.4	0.60	83.9	0.25
CASE-III	90.1	0.36	83.29	0.46	85.3	0.35	90.05	0.21
CASE-IV	95.4	0.06	89.9	0.14	92.2	0.10	93.37	0.08

The accuracy in the baseline model for Hindi and English is quite low because the baseline model cannot capture the half form representation of letters. As there is no half form in

Punjabi, a Hindi name, when spelled in Punjabi, uses the full form of the character instead of its half form. For example, the name इन्द्र (Inder) when spelled in Punjabi will look like ਇੰਦਰ [indar]. Here, the half form of न and र in the Hindi name are represented by (tippi-a character for nasal sound) and ਰ, respectively. When this form is transliterated by the baseline model, it will produce इंदर, which is incorrect. Similar is the case with English and other foreign words. So, due to the character gap in Punjabi and other languages, the word accuracy rate for the baseline is low. It is also interesting to note that when words that are originally from Hindi and used in Punjabi are transliterated back to Hindi, the accuracy rate is lower than other types of words (for Cases III and IV). The reasons are twofold. First, Hindi words written in Punjabi are an approximate transliteration of the original Hindi word in Punjabi. Depending upon the perception of the transliterator, a Hindi word may have more than one representation in Punjabi, *e.g.*, the Hindi name आचार्या (*Acharya*) can be represented in Punjabi as "ਆਚਾਰਿਆ ", " ਆਚਾਰੀਆ ", " ਅਚਾਰੀਆ ", and " ਅਚਾਰਿਆ ". Only the first representation, when again transliterated back into Hindi, converts to the correct representation. Other representations are converted into such strings by baseline and rule based modules that they have no match in the unigram table, making the Soundex module inefficient. Normalization of spellings at the source by a similar Soundex approach may improve the results.

Table 10. Breakdown of Word Accuracy rate and Avg. Lev Dist for person names, location names, and other words according to origin of source language of input word

		Punjabi Words		Hindi Words		English Words		Foreign Words	
		WAR	ALD	WAR	ALD	WAR	ALD	WAR	ALD
CASE-I	Person Name	86.9	0.4	63.92	0.77	79.51	0.59	82.78	0.39
	Location Name	76.61	0.60	67.09	0.48	42.30	1.23	73.68	0.26
	Other Words	87.02	0.27	30	1.3	52.21	0.83	66.66	0.55
CASE-II	Person Name	91.6	0.12	70.49	0.43	84.33	0.32	87.41	0.24
	Location Name	91.12	0.14	75.95	0.34	57.69	0.61	73.68	0.26
	Other Words	91.8	0.11	29.04	1.25	56.63	0.71	66.66	0.33
CASE-III	Person Name	91.4	0.35	83.14	0.14	90.36	0.34	90.72	0.23
	Location Name	77.41	0.56	82.28	0.29	65.38	0.90	94.73	0.05
	Other Words	92.30	0.22	90.1	0.3	87.61	0.23	88.88	0.33
CASE-IV	Person Name	95.8	0.05	89.7	0.12	95.18	0.08	94.7	0.08

	Location Name	92.74	0.08	91.14	0.14	86.53	0.11	94.73	0.05
	Other Words	95.41	0.05	90.8	0.25	92.47	0.10	88.88	0.11
CASE-V	Person Name	91.4	0.32	85.14	0.15	88.36	0.33	89.72	0.21
	Location Name	78.1	0.54	82.8	0.28	67.38	0.85	94.23	0.05
	Other Words	92.6	0.21	91.1	0.31	86.94	0.21	89.28	0.3

The system also benefits from the spelling correction capability of the Soundex method. It can improve the results by selecting the correct output for a given wrong input, *e.g.,* for the input ਅਨੁਪਮ (correct is ਅਨੁਪਮ [anupam]), the Soundex method selects the correct variation अनुपम instead of अनूपम. On the other hand, at times it selects the wrong target word, *e.g.*, for input वैंट [kaiṇṭ] (cantt) the correct output is कैंट but the Soundex method selects केंट as output. Another problem with the Soundex based approach is that it is dependent upon the unigram table. Although this table is generated from a large amount of data, still it is not exhaustive. New terms and names are coined every day, and these need to be included in the unigram table regularly. If the data is not present in this table, then the system fails. Such cases, however, are few and overall accuracy of the system improves with the combined approach of baseline + rule based + Soundex based approach.

7. Conclusion

The results show that following the rule based followed by Soundex approach produces the best results and the words can be transliterated with considerable accuracy. A fully accurate transliteration system is not possible due to the inherent problems, missing corresponding letters in two scripts, conjunct form, *etc*. Although it is possible to transliterate across the scripts preserving the basic sounds of the source language, there will be some variations because the word in the source script may be pronounced somewhat differently in the target script, as per local conventions in each region. This results in more time required by the users to interpret the word in the context of original language from which the word comes. The transliterated text will be correctly understood only if the reader has knowledge of the conventions used for representing sounds in the source script. The aim of the module at least is to present a nearest possible transliterated text to the user instead of going blank to such words. This module is successfully used for transliterating proper nouns in Punjabi to Hindi machine translation system and in the Cross Language Information Retrieval system "PunjabiKhoj," which is a search engine for Punjabi language.

Although the system performs well, still there is room for improvement. Transcription rules based upon IPA and statistical models will be considered in the future. Testing of the

system on precision, recall, and F-score is on our future agenda.

Reference

Al-Onaizan, Y., & Knight, K. (2002). Translating named entities using monolingual and bilingual resources. *Proceedings of the 40th ACL*, Philadelphia, 400-408.

Arnold, D., Balkan, L., Humphreys, R. Lee, Meijer, S., & Sadler, L. (1995). *Machine Translation: An Introductory Guide*. NCC Blackwell, Manchester, Oxford.

Balkan, L. (1998). Test suites: some issues in their use and design. In *proceedings of the International conference* on "Machine translation: ten years on" held at Cranfield University, England, 12-14 November 1994 (Cranfield University Press, 1998).

Bhatia, T. K. (1993). Punjabi: A Cognitive-descriptive Grammar. *Descriptive Grammars,* Routledge, London.

Chopde, A. (2001). *Printing Transliterated Indian Language Documents*, ITRANS, from website http://www.aczoom.com/itrans/idoc/idoc.html (Accessed on 22 Aug 2006).

Elliott, D., Hartley, A., & Atwell, E. (2003). Rationale for a multilingual aligned corpus for machine translation evaluation. In *International Conference on Corpus Linguistics (CL2003)*, 191-200. Lancanster, UK.

Estrella, P., Hamon, O., & Popescu-Belis, A. (2007). How much data is needed for reliable MT evaluation? Using bootstrapping to study human and automatic metrics. *MT Summit XI*, 10-14 September 2007, Copenhagen, Denmark. *Proceedings*, 167-174.

Goyal, V., & Lehal, G. S. (2009). Hindi-Punjabi Machine Transliteration System (For Machine Translation System). George Ronchi Foundation Journal, Italy, 64(1), 2009.

Hoard, J. (1991). Preliminaries to the Development of Evaluation Metrics for Natural Language Semantic and Pragmatic Analysis Systems. in Neal, J.G. and Walter, S.M. (eds.), (1991): *Natural Language Processing Systems Evaluation Workshop*, Report RLTR- 91-362, Rome Laboratory.

Jung, S. Y., Hong, S. L. & Paek, E. (2000). An English to Korean Transliteration Model of Extended Markov Window. *Proceedings of COLING 2000*.

KalyanaSundram, K. *Mylai Tamil Font for preparation of Texts in Tamil Script on Computers*, from website http://tamilelibrary.org/teli/mylai1.html.

Kang, B.J. & Choi, K.-S. (2000). Automatic Transliteration and Back-transliteration by Decision Tree Learning. *Proceedings of the 2nd International Conference on Language Resources and Evaluation*, Athens, Greece.

Kanneganti, R. & Kishore, A. *Rice Inverse Transliterator (RIT) Software*, from website http://www.teluguworld.org/RIT/rit.html.

Knight, K. & Graehl, J. (1998). Machine Transliteration. *Computational Linguistics*, 24(4).

Lee, C. & Chang, J. S. (2003). Acquisition of English-Chinese Transliteration Word Pairs from Parallel-Aligned Texts using a Statistical Machine Translation Model. *Proceedings*

of HLT-NAACL Workshop: Building and Using parallel Texts Data Driven Machine Translation and Beyond, 2003, Edmonton, 96-103.

Malik, M.G.A. (2006). Punjabi Machine transliteration. *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, 1137-1144.

Meng, H. M., Lo, W., Chen, B., & Tang, K. (2001). Generate Phonetic Cognates to Handle Name Entities in English-Chinese cross-language spoken document retrieval. *Proceedings of ASRU,* 2001.

Odell, M.K., & Russel, R.C. (1973). US Patent 1261167, 1918, cited in Knuth (1973) "Sorting and Searching: The art of computer programming" Volume 3, Addison Wesley, reading, MA.

Oh, J.H., & Choi, K.S. (2002). An English-Korean transliteration model using pronunciation and contextual rules. *Proceedings of the 19th international conference on Computational linguistics,* 1, 1-7.

Saini, T. S., & Lehal, G. S. (2008). Shahmukhi to Gurmukhi Transliteration System: A Corpus based Approach. *Research in Computing Science* (Mexico), 33, 151-162.

Srinivasan. (1995). "ADHAWIN", in "Transliteration schemes" from website http://acharya.iitm.ac.in/multi_sys/transli/schemes.php.

The Devanagari Script Block. (2008). from website http://people.w3.org/rishida/uniprop32/descn-devanagari.html accessed on 7 Feb 2008.

Transliteration Principles. (2008). from http://acharya.iitm.ac.in/multi_sys/translit.php accessed on 7 Feb 2008.

Virga, P., & Khudanpur, S. (2003). Transliteration of proper names in cross-language applications. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval*, 365-366.

Viswanadha, R. (2002). Transliteration of Tamil and Other Indic Scripts. *Tamil Internet 2002*, California, USA.

Wan, S., & Verspoor, C. M. (1998). Automatic English-Chinese name transliteration for development of multilingual resources. *Proceedings of COLING-ACL'98.*

Writing Systems. (2010). form website http://www.omniglot.com/writing/alphabets.htm.

Zhang, Y., & Vogel, S. (2004). Measuring Confidence Intervals for the Machine Translation Evaluation Metrics. In I*nternational Conference on Theoretical and Methodological Issues in Machine Translation (TMI 2004)*. Baltimore, MD.

Computational Linguistics and Chinese Language Processing
Vol. 15, No. 2, June 2010, pp. 103-126

A Posteriori Individual Word Language Models for Vietnamese Language

Le Quan Ha[*], Tran Thi Thu Van[*], Hoang Tien Long[+],

Nguyen Huu Tinh[+], Nguyen Ngoc Tham[+], and Le Trong Ngoc[*]

Abstract

It is shown that the enormous improvement in the size of disk storage space in recent years can be used to build individual word-domain statistical language models, one for each significant word of a language that contributes to the context of the text. Each of these word-domain language models is a precise domain model for the relevant significant word; when combined appropriately, they provide a highly specific domain language model for the language following a cache, even a short cache. Our individual word probability and frequency models have been constructed and tested in the Vietnamese and English languages. For English, we employed the Wall Street Journal corpus of 40 million English word tokens; for Vietnamese, we used the QUB corpus of 6.5 million tokens. Our testing methods used *a priori* and *a posteriori* approaches. Finally, we explain adjustment of a previously exaggerated prediction of the potential power of *a posteriori* models. Accurate improvements in perplexity for 14 kinds of individual word language models have been obtained in tests, (i) between 33.9% and 53.34% for Vietnamese and (ii) between 30.78% and 44.5% for English, over a baseline global tri-gram weighted average model. For both languages, the best *a posteriori* model is the *a posteriori* weighted frequency model of 44.5% English perplexity improvement and 53.34% Vietnamese perplexity improvement. In addition, five Vietnamese *a posteriori* models were tested to obtain from 9.9% to 16.8% word-error-rate (WER) reduction over a Katz trigram model by the same Vietnamese speech decoder.

[*] Faculty of Information Technology, Hochiminh City University of Industry, Ministry of Industry and Trade, 12 Nguyen Van Bao, Ward 4, Go Vap District, Hochiminh City, Vietnam
E-mail: lequanha@ hui.edu.vn; NLp.Sr@ Shaw.ca; letrongngoc@ hui.edu.vn

[+] Faculty of Information Technology, Nong Lam University of Hochiminh City, Block 6, Linh Trung Ward, Thu Duc District, Hochiminh City, Vietnam
E-mail: {06130155, 06130204, 06130194}@ st.hcmuaf.edu.vn

Keywords: *A Posteriori*, Stop Words, Individual Word Language Models, Frequency Models.

1. Introduction

A human is able to work out the precise domain of a spoken sentence after hearing only a few words. The clear identification of this domain makes it possible for a human to anticipate the following words and combination of words, thus, recognizing speech even in a very noisy environment. This ability to anticipate still cannot be replicated by statistical language models. In this paper, we suggest one way that significant improvement in language modeling performance can be achieved by building domain models for significant words in a language. The word-domain language model extends the idea of cache models (Kuhn & De Mori, 1990) and trigger models (Lau, Rosenfeld, & Roukos, 1993) by triggering a separate *n*-gram language model for each significant word in a cache and combining them to produce a combined model.

The word-domain models are built by collecting a training corpus for each significant word. This is done by amalgamating the text fragments where the word appears in a large global training corpus. In this paper, the text fragments are the sentences containing the significant words. Sicilia-Garcia, Ming, and Smith (2001, 2002) have shown that larger fragments are not needed. We define a significant word as any word that significantly contributes to the context of the text or any word that is (i) not a stop word, *i.e*. not an article, conjunction, or preposition; (ii) not among the most frequently used words in the language, such as “will”; and (iii) not a common adverb or adjective, “now,” “very,” “some,” *etc*.

All other words are considered significant, and a corpus is built for each. A statistical language model is then calculated from this corpus, *i.e*. from all of the sentences containing the word. Therefore, the model should be able to represent the domain of that word. This approach entails a very large number of individual word language models being created, which requires a correspondingly large amount of disk storage; previous experiments by Sicilia-Garcia, Ming, and Smith (2005) were done on twenty thousand individual word language models, which occupied approximately 180 GigaBytes. Thus, this tactic is feasible only given the relatively recent increase in affordable hard disk space. These word models gradually developed from the PhD work of Sicilia-Garcia 1999-2005. Almost at the same time as her research, similar research work was done by Blei, Ng, and Jordan (2003); they originally started from the ideas of Hofmann, Puzicha, and Buhmann (1998) and from Hofmann (1999).

The remaining sections are organized in the following way. First, we discuss the weighted average model our developed approach lies upon. Then, we discuss both the probability models - including linear interpolation and the exponential decay model - and the

weighted models, such as the weighted probability model, weighted exponential model, and linear interpolation exponential model with weights. Then, we discuss their corresponding frequency models and we discuss *a priori* and *a posteriori* testing methods. Following this, the corpus, experiments, and results are shown for the probability models, for the frequency models, and for *a posteriori* models. Finally, we provide some conclusions.

2. The Language Models

Experiments had shown that we needed to combine the global language model with the individual word-domain models in order to obtain good results. (This may be due to the limited size of the global corpus in our tests, which was 40 million tokens.) So, we first built a language model for the whole global corpus. Frequencies of words and phrases derived from the corpus and the conditional probability of a word given a sequence of preceding words were calculated. The conditional probabilities were approximated by the maximum likelihoods:

$$P_{ML}\left(w_i \middle| w_1^{i-1}\right) = \frac{f\left(w_1^i\right)}{f\left(w_1^{i-1}\right)} = \frac{f\left(w_1 \ldots w_{i-1} w_i\right)}{f\left(w_1 \ldots w_{i-1}\right)} \tag{1}$$

where $f\left(w_1^n\right)$ is the frequency of the phrase $w_1^n = w_1 \ldots w_{n-1} w_n$ in the text. These probabilities were smoothed by one of the well-known methods, such as Turing-Good estimation (Good, 1953) or the Katz back-off method (Katz, 1987). Although any of these could be used in our experiment to demonstrate the principle of our multiple word-domain models, it was convenient to use the empirical weighted average (WA) linear interpolation *n*-gram model (O'Boyle, Owens & Smith, 1994) because of its simplicity. It gives results comparable to the Katz back-off method but is much quicker to use. The weighted average probability of a word w given the preceding words $w_1 \ldots w_{m-1} w_m$ is:

$$P_{WA}\left(w \middle| w_1^m\right) = \frac{\mu_0 P_{ML}\left(w\right) + \sum_{i=1}^{m} \mu_i P_{ML}\left(w \middle| w_{m+1-i}^m\right)}{\sum_{i=0}^{m} \mu_i} \tag{2}$$

where the weighted functions (in the simplest case) are given by

$$\mu_0 = Ln\left(T\right) \quad and \quad \mu_i = Ln\left(f\left(w_{m+1-i}^m\right)\right).2^i \tag{3}$$

where T is the number of tokens in the corpus and $f\left(w_{m+1-i}^m\right)$ is the frequency of the sentence $w_{m+1-i} \ldots w_m$ in the text. The unigram maximum likelihood probability of a word is:

$$P_{ML}\left(w\right) = \frac{f\left(w\right)}{T} \tag{4}$$

The language model defined by Equations (2) and (4) is called the global language model when trained on the global corpus. The creation of a language model for each significant word is formed in the same manner as the global language model.

3. Probability Models

We need to combine the probabilities obtained from each word-domain language model and from the global language model in order to obtain a combined probability for a word, given a sequence of words. One simple way to do this is a mathematical combination of the global language model and the word language models in a linear interpolated expression as:

$$P\left(w\middle|w_1^n\right)=\lambda_G P_G\left(w\middle|w_1^n\right)+\sum_{i=1}^{m}\lambda_i P_i\left(w\middle|w_1^n\right) \tag{5}$$

where $\lambda_G+\sum_{1}^{m}\lambda_i=1$ and $P_G\left(w\middle|w_1^n\right)$ is the conditional probability of the word w following a phrase $w_1...w_{n-1},w_n$ in the global language model, P_i is the conditional probability in the word language model for the significant word w_i, λ_i is the correspondent weight, and m is the number of word models that are included. Ideally, the λ_i parameters would be optimized using a held-out training corpus; however, this is not practical as we do not know which combination of words w_i will arise in the cache. So, a simpler approach is needed.

3.1 Linear Interpolation

A simple method of choosing the λ-values is to give the same weight to all of the word language models but a different weight to the global language model and to put a restriction on the number of word language models to be included. This weighted model is defined as

$$P\left(w\middle|w_1^n\right)=\lambda.P_G\left(w\middle|w_1^n\right)+\frac{(1-\lambda)}{m}\left[\sum_{i=1}^{m}P_i\left(w\middle|w_1^n\right)\right] \tag{6}$$

and λ and m are parameters that are chosen to optimize the model.

3.2 Exponential Decay Model

A method was developed based on an exponential decay of the word model probabilities with distance since a word appearing several words before the target word will generally be less relevant than more recent words. Given a sequence of Vietnamese words, for example, “Tôi đã tri giao với HUI” (meaning “I had friendly relations with HUI”) in Table 1, where 5, 4, 3, 2, 1 represent the distance of the word from the target word “HUI”. The words “tri” (relation) and “giao” (friendly) are significant words for which we have individual word language models.

Table 1. An explanation of distance of words.

Tôi	đã	**tri**	**giao**	với	**HUI**
5	4	3	2	1	

This model for the word w, where w represents the significant word "HUI," is as follows:

$$P\left(w\middle|w_1^n\right)=\frac{P_G\left(w\middle|w_1^n\right)+P_{tri}\left(w\middle|w_1^n\right).\exp\left(-3/d\right)+P_{giao}\left(w\middle|w_1^n\right).\exp\left(-2/d\right)}{1+\exp\left(-3/d\right)+\exp\left(-2/d\right)} \tag{7}$$

where $P_G\left(w\middle|w_1^n\right)$ is the conditional probability of the word w following a phrase $w_1,w_2\ldots w_n$ in the global language model and $P_{tri}\left(w\middle|w_1^n\right)$ is the conditional probability of the word w following a phrase $w_1\ldots w_{n-1}w_n$ in the word language model for the significant word "tri". The same definition applies for the word model "giao". d is the exponential decay distance with d=5, 10, 15, *etc*. A cache or cut-off is introduced in the model

if $l \geq cache$ => replace exp(-l/d) by 0

where l is the distance from the significant word to the target word.

3.3 Weighted Models

In the two methods above, the weights for the word language models were independent of the size of the word training corpora or the global training corpus. So, we introduced new weights to these models that depend on the size of the training corpora. These weights are functions of the size of the word training corpora, *i.e*. the number of tokens of the training corpora T_i. Examples of the weights can be seen in Table 2.

Table 2. Some of the weights in weighted models.

Weights
$Ln(1+LnT_i)$
$Sqrt(LnT_i)$
LnT_i
$Sqrt(T_i)$
T_i/LnT_i
T_i
T_iLnT_i

An obvious weight to use is a log function to match the information theory, but other weights were also tried. All of these weights are studied in order of weight difference as T_i increases, from the largest difference to the smallest. Their relationship can be seen in Figure 1.

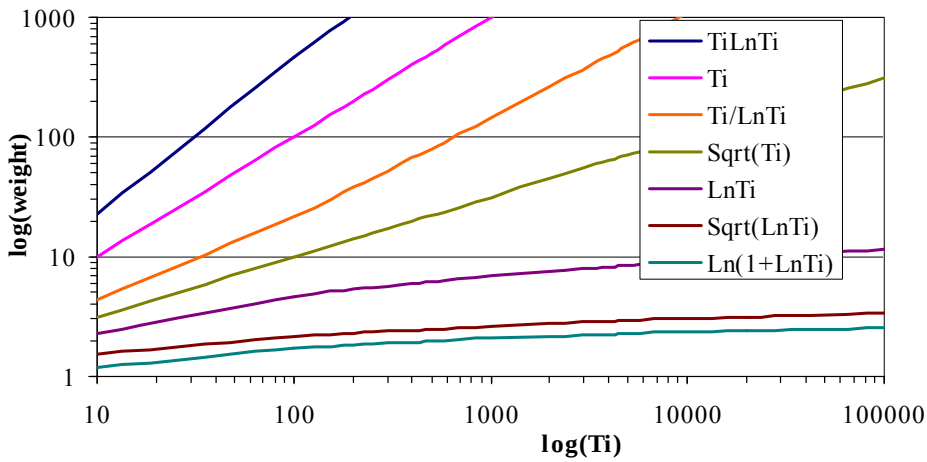


Figure 1. Value of the weights used in these models.

3.3.1 Weighted Probability Model

The weighted probability model is based on the idea that the weight given to a word language model should depend on the size of the training corpora. It is described as follows:

$$P\left(w\middle|w_1^n\right)=\frac{\beta_G.P_G\left(w\middle|w_1^n\right)+\sum_{i=1}^{m}\beta_i.P_i\left(w\middle|w_1^n\right)}{\beta_G+\sum_{i=1}^{m}\beta_i} \tag{8}$$

where β_G is the weight for the global language model and β_i is the weight for the word model for the word w_i. We give more weight to those word models with a small training corpus as they represent models for the less frequent words, which have the most information. The weights used are functions of the size of the word training corpora in Table 2, that is, of the number of tokens of the training corpora T_i.

3.3.2 Weighted Exponential Model

The weighted exponential language model is a combination of the weighted probability model and the exponential decay model. Each language model has two functions: one is the exponential decay, in terms of the distance from the significant word, and the second function is a weight, depending on the size of the word training corpora. We define this model as:

$$P\left(w\middle|w_1^n\right)=\frac{\beta_G.P_G\left(w\middle|w_1^n\right)+\sum_{i=1}^{m}\beta_i.\exp\left(-\frac{x_i}{d}\right)P_i\left(w\middle|w_1^n\right)}{\beta_G+\sum_{i=1}^{m}\beta_i.\exp\left(-\frac{x_i}{d}\right)} \tag{9}$$

where x_i is the distance from the word w_i model.

3.3.3 Linear Interpolation Exponential Model with Weights

Finally, we decided to try another model that is based on a combination of all previous methods. Based on the idea that the global language model could be weighted in a different way from the word language models, the probability of a word given the previous words is:

$$P\left(w\middle|w_1^n\right)=\lambda P_G\left(w\middle|w_1^n\right)+(1-\lambda)\frac{\sum_{i=1}^{m}\beta_i.\exp\left(-\frac{x_i}{d}\right)P_i\left(w\middle|w_1^n\right)}{\sum_{i=1}^{m}\beta_i.\exp\left(-\frac{x_i}{d}\right)} \tag{10}$$

This is equivalent to combining all of the methods seen previously into one model, which we call the linear interpolation exponential model with weights.

4. Frequency Models

Instead of combining probabilities to obtain a dynamic language model, it is also possible to combine frequencies before calculating probabilities, *i.e.* a revised maximum likelihood. To do this, replace Equation (1) with:

$$P_{ML}(w_i\mid w_1^{i-1})=\frac{\lambda_G f_G(w_1^i)+\sum_{i=1}^{m}\lambda_i f_i(w_1^i)}{\lambda_G f_G(w_1^{i-1})+\sum_{i=1}^{m}\lambda_i f_i(w_1^{i-1})} \tag{11}$$

This can then be combined using the WA model in Equation (2). This simple method is automatically normalized, and it is easy to implement and fast to execute. The choice of λ is still critical but cannot be optimized from a held out corpus. For the frequency model, we also combine the frequencies using the same methods that are used for probabilities.

4.1 Linear Interpolation Frequency Model

The linear interpolation model applied to the frequency is defined as:

$$f_{FM}\left(w_1^n\right)=f_{FM}\left(w_1w_2...w_n\right)=\lambda.f_G\left(w_1^n\right)+\frac{(1-\lambda)}{m}\left[\sum_{i=1}^{m}f_i\left(w_1^n\right)\right]\quad if\ m>0 \tag{12}$$

where $f_G\left(w_1^n\right)$ is the frequency of the phrase $w_1, w_2 \ldots w_n$ from the global model, $f_i\left(w_1^n\right)$ is the frequency of the phrase $w_1, w_2 \ldots w_n$ from the word w_i model, and m and λ>0 are chosen

parameters.

4.2 Exponential Decay Frequency Model

A method was used based on an exponential decay of the frequencies:

$$f_{FM}\left(w_1^n\right) = f_{FM}\left(w_1 w_2 ... w_n\right) = f_G\left(w_1^n\right) + \sum_{i=1}^{m} \exp\left(-\frac{x_i}{d}\right) f_i\left(w_1^n\right) \tag{13}$$

4.3 Weighted Models

We introduce new weights to these models; they are functions shown in Table 2.

4.3.1 Weighted Frequency Model

The weighted frequency model is a mathematical combination of the word language models with the global language models. The frequency for this model is defined as follows:

$$f_{FM}\left(w_1^n\right) = f_{FM}\left(w_1 w_2 ... w_n\right) = \beta_G f_G\left(w_1^n\right) + \sum_{i=1}^{m} \beta_i . f_i\left(w_1^n\right) \tag{14}$$

4.3.2 Weighted Exponential Decay Frequency Model

As can be seen from the previous section, the frequencies are weighted depending on the size of the training corpora only. In this model, the exponential decaying factor is added to these weights. The new weighted exponential frequency becomes:

$$f_{FM}\left(w_1^n\right) = f_{FM}\left(w_1 w_2 ... w_n\right) = \beta_G f_G\left(w_1^n\right) + \sum_{i=1}^{m} \beta_i . \exp\left(-\frac{x_i}{d}\right) f_i\left(w_1^n\right) \tag{15}$$

5. Testing Methods

Perplexity is a well known measure of the performance of a language model (Jelinek, Mercer, & Bahl, 1983). We calculate the perplexity of each sentence, w_1^n, by

$$P(w_1^n) = P(w_1)P(w_2 \mid w_1)...P(w_n \mid w_1^{n-1}) \tag{16}$$

and the perplexity of a sentence by

$$PP(w_1^n) = \exp\left(-\frac{1}{m}\sum_{i=1}^{m} Ln\left(P(w_i \mid w_1 w_2 ... w_{i-1})\right)\right) \tag{17}$$

where *m* is the number of words in the sentence.

There are two methods of calculating the constituent probabilities on the right hand side of Equation (17) using the word domain language models, one *a priori* by Sicilia-Garcia *et al.* (2001, 2002) and the second *a posteriori,* which was first tried by Sicilia-Garcia *et al.* (2005.)

5.1 *A Priori* Method

In the *a priori* method, we use the global language model and (possibly) individual word models from earlier sentences (*i.e.* from the cache) at the beginning of the sentence because we do not know which significant words are going to appear in the sentence. We then add in a word language model for each significant word after it appears in the sentence. Thus, in the sentence, "The cat sat on the mat," neglecting previous sentences, the first two words are modeled using the global language model, the probability P(sat| the cat) is calculated using the global model combined with the word model for "cat," and the last three words are modeled using the global model combined with the word models for "cat" and "sat".

5.2 *A Posteriori* Method

This, however, is not the only way in which models are tested. For example, in domain language models, researchers extract a whole sentence, paragraph, or document from the test file, find all of the significant words within it, and use all of these words to perform an optimization of the possible domains to find the domain or combination of domains to minimize the perplexity (Seymore, Chen, & Rosenfeld, 1998; Donnelly, 1998; Iyer & Ostendorf, 1999; Sicilia-Garcia *et al.*, 2005.)

To make comparisons with these other domain methods, Sicilia-Garcia *et al.* (2005) also tried this *a posteriori* method to calculate perplexity for word-domain models. To do this, they extracted all of the significant words in a sentence and built a language model based on the global model and the word domain models for the significant words. This was then used to calculate the perplexity of the whole sentence. In the example above, "The cat sat on the mat," the perplexity was calculated using the global model combined with the word domain models for the three words "cat," "sat," and "mat" for the whole sentence calculation. Using this approach, they obtained 68%-69% improvement in perplexity when using the *a posteriori* weighted probability model and the *a posteriori* weighted frequency model. They accepted that this 69% improvement exaggerated the performance but showed the potential power of these *a posteriori* models. Upon further analysis of this, however, we have discovered a slight flaw in the calculation. This can be demonstrated by again considering the example above. They calculated the *a posteriori* probability P(sat| the cat) by employing the global model, the word "cat" model, the word "sat" model, and the word "mat" model. As the unigram "sat," as well as its *n*-grams, obviously occur themselves in every sentence of its own target word "sat" model, this yields an unnaturally high probability P_{sat}(sat| the cat). Hence, the flaw: if we replace the word "sat" by any significant words *xxx* in the test, the probability P_{xxx}(*xxx*| the cat) will also obtain unnaturally high values in its own word *xxx* model.

In this work, we propose and test a corrected method for calculating the *a posteriori* probability P(sat| the cat), which uses only the global model and the word models for "cat"

and "mat," excluding the target word "sat" model. Furthermore, we have tested three different ways of combining word models in our calculation for the linear interpolation probability, the exponential decay probability, and the weighted probability models in the whole sentence test:

i. applying the word models appearing from the beginning until the end of each sentence, excluding the target word model

ii. applying the word models within the phrase history first then the word models that appear later in each sentence, and

iii. ignoring the order of appearance of significant words in each sentence, either existing in the phrase history or occurring later, but applying those significant word models that are located from nearer to farther distances, relative to the target word.

We find out that the (iii) calculation provides 1%-3% better perplexity improvements than (i) and (ii) calculations for all three models. This means that nearer word models supply more reliable probabilities.

Sicilia-Garcia (2002) suggested another whole paragraph calculation with poorer results.

6. Corpus

The methods described above were compared in some Vietnamese and English experiments. For English, we used the Wall Street Journal (WSJ) corpus (Paul & Baker, 1992). Previous research by Sicilia-Garcia *et al.* (2001, 2002, 2005) displayed how the individual word probability models depend on the size of the training corpus for two subsets of the WSJ of approximately 16 million (1988) and 6 million words (1989). The well-known WSJ test file (Paul *et al.*, 1992) contains 584 paragraphs, 1,869 sentences, 34,781 tokens, and 3,677 word types. In this work, we develop these models for the whole combined WSJ corpus of 40 million words. The results reveal a lower perplexity for the larger 40 million word corpus, compared to Sicilia-Garcia *et al.*

For Vietnamese, we employ the syllabic Vietnamese corpus by Ha in 2002 (http://hochiminhcityuniversityofindustry-lequanha.schools.officelive.com/VietnameseQUBCorpus.aspx), with a size of 6,564,445 tokens and 31,402 types. This corpus was collected from popular Vietnamese newspapers, such as The Youth Newspaper, Saigon Liberation Newspaper, Vietnamese Culture and Sports, Motion Pictures Magazine, and Vietnam Express, along with traditional and modern novels and short stories.

Vietnamese is the national and official language of Vietnam. It is the mother tongue of 86% of Vietnam's population, and of about three million overseas Vietnamese people. It is also spoken as a second language by many ethnic minorities of Vietnam. It is part of the Austro-Asiatic language family, of which it has its most speakers, having several times more speakers than the other Austro-Asiatic languages put together. Some Vietnamese vocabulary

has been borrowed from Chinese, and it was formerly written using a Chinese-like writing system, albeit in a modified format and with vernacular pronunciation. As a byproduct of French colonial rule, the language displays some influence from French, and the Vietnamese writing system in use today is an adapted version of the Latin alphabet, with additional diacritics for tones and certain letters.

Vietnamese is basically a monosyllabic language with six tones, which gives the language a sing-song effect. A syllable can be repeated with any one of six tones to indicate six different meanings. For example, the syllable "ma" has six different meanings according to the tone this Vietnamese syllable carries - with *level* tone, "ma" means "phantom" or "ghost"; with *low* tone, "mà" means "but," "which," or "who"; with *high rising glottalized* tone, "mã" means "code"; with *dipping-rising* tone, "mả" means "tomb"; with *high rising* tone, "má" means "cheek"; and with *low glottalized* tone, "mạ" means "young rice seedling".

Due to the semantic impact of tonality, we would like to apply our language models for Vietnamese syllables instead of English words.

We also established our Vietnamese test text from the above Vietnamese newspapers, but its content was taken from newspapers of the year 2008, much later than Ha's training text in 2002. Therefore, the Vietnamese test text is totally different from Ha's corpus; it includes 33,108 Vietnamese syllable tokens of 2,226 syllable types within 3,321 sentences.

7. Results

We will first present the results for each *a priori* model, starting from the probability models and moving to the frequency models then show our *a posteriori* results. All perplexity values shown in the tables are accompanied with a percentage, which shows the improvement compared to the global base-line trigram WA model.

7.1 Results for Probability Models

We present the results for all *a priori* probability models, starting from the linear interpolation probability model and going to the linear interpolation exponential decay probability model with weights. It can be seen from the results that the best English and Vietnamese probability model is the weighted exponential probability model, with overall results in Tables 3 and 4.

The best performance achieved using each different type of probability models is shown in Table 5 and Table 6.

The best model is the weighted exponential probability model, with 34% improvement for English and 37% for Vietnamese, while the linear interpolation exponential model with weights - our combination of all of the models - disappointingly only improves 32% for English and 33.9% for Vietnamese. For these models, the number of individual word models

required in the cache to reach the maximum performance is 16-23 English words and 29-64 Vietnamese syllables. So, the individual word-domain language model reduces the size of the cache needed from 500 words, as in other models (Clarkson & Robinson, 1997; Donnelly, Smith, Sicilia-Garcia & Ming, 1999), to less than 30 English words or 64 Vietnamese syllables, which is important for spoken language and is closer to the ability of humans.

In Table 5 and Table 6, *WM* represents the number of word language models that is *m* in Equations (6), (8), (9), and (10).

Table 3. The weighted exponential model (English WSJ.)

	Sentence/WSJ		
***n*-gram**	**Perplexity**	**Improvement**	***(d, Cache, Function)***
tri-gram	62.71	16.78%	*(8, 75, Sqrt(1/LnT$_i$))*
5-gram	51.09	32.21%	*(8, 70, LnT$_i$)*
7-gram	49.91	33.77%	*(7, 75, LnT$_i$)*
9-gram	**49.82**	**33.90%**	***(7, 75, LnT$_i$)***

Table 4. The weighted exponential model (Vietnamese QUB.)

	Sentence/QUB		
***n*-gram**	**Perplexity**	**Improvement**	***(d, Cache, Function)***
tri-gram	94.70	22.98%	*(13, 100, Sqrt(LnT$_i$))*
5-gram	80.16	34.81%	*(13, 100, LnT$_i$)*
7-gram	78.12	36.47%	*(13, 100, LnT$_i$)*
9-gram	**77.57**	**36.92%**	***(13, 100, LnT$_i$)***

Table 5. Improvement in perplexity for different probability models, all in sentence contexts (English WSJ.)

Models	**tri-gram**	**9-gram**	**Best Values**
Global	0.00%	26.77%	
Linear Interpolation	11.16%	31.98%	*λ=0.7, WM=23*
Exponential Decay	16.75%	33.72%	*Decay=6, Cache=70*
Weighted Probability	13.90%	32.52%	*WM=16, Sqrt(T$_i$)*
Weighted Exponential	**16.78%**	**33.90%**	***Decay=7, Cache=75, LnT$_i$***
Linear Interpolation Exponential with Weights	12.91%	32.28%	*λ=0.6, Decay=13, Cache=65, Ln(1+LnT$_i$)*

Table 6. Improvement in perplexity for different probability models, all in sentence contexts (Vietnamese QUB.)

Models	tri-gram	9-gram	Best Values
Global	0.00%	20.88%	
Linear Interpolation	18.46%	34.02%	*λ=0.6, WM=64*
Exponential Decay	22.88%	36.54%	*Decay=10, Cache=100*
Weighted Probability	20.45%	34.45%	*WM=29, Sqrt(T_i)*
Weighted Exponential	**22.98%**	**36.92%**	***Decay=13, Cache=100, LnT$_i$***
Linear Interpolation Exponential with Weights	18.40%	33.88%	*λ=0.6, Decay=57, Cache=100, 1/Ln(1+LnT$_i$))*

Our weighted exponential model is a special case of the linear interpolation exponential with weights. Hence, it should not have better performance. Sicilia-Garcia *et al.* (2001, 2002, 2005) were also disappointed when this combination model of all other models was not good. The reason for this unusual observation is a conflict occurring between the weighted exponential model (that is a combination of exponential decay and weighted probability models) and the linear interpolation model. The linear interpolation model was optimized on the condition that all significant words are equally treated, while the weighted exponential model treats significant words differently from each other.

In the linear interpolation model, the global "weight" is λ = 0.6 and each Vietnamese syllabic model equally optimized at a "weight" of $(1-\lambda)/m$ = (1-0.6)/64 = 0.006,25. For the weighted exponential model, however, the global weight is Ln(40M) = 17.5 and a Vietnamese syllabic model with size 10,000 has a weight of Ln(10,000) = 9.21. Another Vietnamese syllabic model with size 100 has its weight of Ln(100) = 4.61. On the linear interpolation exponential with weights, when these weights are multiplied or interpolated together, they break the optimization of both the linear interpolation model and the weighted exponential model; the too small individual "weight" 0.006,25 in the first model and the much larger global weight 17.5 in the latter model are not satisfied.

7.2 Results for Frequency Models

We present the results for each *a priori* frequency model, starting from the linear interpolation frequency model and going to the weighted frequency model. The best English frequency model is the weighted frequency model, and our results for this model are displayed in Table 7. An improvement of 38% has been achieved for the English weighted frequency model.

The best Vietnamese one is the exponential decay frequency model with 47.3% perplexity improvement, as shown in Table 9.

Table 7. The weighted frequency model (English WSJ.)

	Sentence/WSJ		
***n*-gram**	**Perplexity**	**Improvement**	***(WM, Function)***
tri-gram	58.12	22.87%	*(29,1/T_i*Ln(T_i))*
5-gram	47.36	37.16%	*(29, 1/T_i)*
7-gram	**46.70**	**38.03%**	***(29, Ln(T_i)/T_i)***
9-gram	46.73	38.00%	*(29, Ln(T_i)/T_i)*

For the English weighted frequency model, it is important to notice that the perplexity result for the 9-gram model is poorer than the one for the 7-gram language model. We think this is because the word language models in many cases are so small that the 9-gram frequencies are usually zero. In order to recognize or understand a rather short phrase as a tri-gram, its meaning largely depends on its context. The other possibility is that the historical word language models need more weight than the large global model of 40 million tokens. If a significant word language model has 1,000 words in its corpus, then the global weight will approximately be 1/(40M*Ln(40M)) = 1.428E-09 while that word model's weight 1/(1,000*Ln(1,000)) = 0.000,144 is much larger in comparison.

Nevertheless, for longer phrases, such as 5-grams or 9-grams, the meanings of a very long 9-gram is almost obvious by itself and its meaning is less impacted by significant words surrounding it. Sometimes, people understand a long spoken phrase with 9 continuous words without confusion even though they did not hear the previous significant words. Therefore, for longer English phrases, the global weight should gain importance and increase its value, relative to significant words. This means that, in Table 7, the WM = 29 for all n-grams but, with the tri-gram weight $1/(T_i*\text{Ln}(T_i))$, is smaller than the 5-gram weight $1/T_i$, and $1/T_i$ is also smaller than $\text{Ln}(T_i)/T_i$ of 7-grams and 9-grams.

This is not happening in the Vietnamese weighted frequency model in Table 8 because the value of WM is very large, 67. The weight $1/T_i$ is applied to this Vietnamese model and to the corresponding 5-gram English model in Table 7. For example, we consider the following Vietnamese syllabic 9-gram " Tiếng Việt Nam là một ngôn ngữ đánh vần ," its closest English phrase - "Vietnamese is a syllabic language" - is only a word 5-gram. Therefore, because many English 5-grams will correspond to Vietnamese syllabic 7-grams or 9-grams, this Vietnamese model has the same weights as the 5-gram English weighted frequency model.

Table 8. The weighted frequency model (Vietnamese QUB.)

	Sentence/QUB		
***n*-gram**	**Perplexity**	**Improvement**	***(WM, Function)***
tri-gram	85.88	30.15%	*(67, $1/T_i$)*
5-gram	73.55	40.18%	*(67, $1/T_i$)*
7-gram	72.63	40.93%	*(67, $1/T_i$)*
9-gram	**72.47**	**41.06%**	***(67, $1/T_i$)***

Table 9. The exponential decay model (Vietnamese QUB.)

	Sentence/QUB		
***n*-gram**	**Perplexity**	**Improvement**	***(Decay, Cache)***
tri-gram	100.60	18.19%	*(150, 150)*
5-gram	71.68	41.71%	*(150, 145)*
7-gram	66.11	46.24%	*(150, 145)*
9-gram	**64.77**	**47.32%**	***(150, 145)***

The best performance of the frequency models is shown by Table 10 and Table 11.

Table 10. Improvement in perplexity for frequency models, all in sentence contexts (English WSJ.)

Models	**tri-gram**	**9-gram**	**Best Values**
Global	0.00%	26.77%	
Linear Interpolation	15.77%	34.32%	*λ=0.003, WM=29*
Exponential Decay	8.13%	30.78%	*Decay=150, Cache=115*
Weighted Frequency	**22.87%**	**38.00%**	***WM=29, $Ln(T_i)/T_i$***
Weighted Exponential Decay	22.84%	37.95%	*Decay=100, Cache=85, $1/T_i$*

Table 11. Improvement in perplexity for frequency models, all in sentence contexts (Vietnamese QUB.)

Models	**tri-gram**	**9-gram**	**Best Values**
Global	0.00%	20.88%	
Linear Interpolation	22.72%	36.20%	*λ=0.002, WM=46*
Exponential Decay	**18.19%**	**47.32%**	***Decay=150, Cache=145***
Weighted Frequency	30.15%	41.06%	*WM=67, $1/T_i$*
Weighted Exponential Decay	30.10%	41.05%	*Decay=100, Cache=100, $1/T_i$*

In Table 10 and Table 11, the best model in the Vietnamese corpus is the exponential decay model, which is different from the best in the English corpus because the Vietnamese language is formed by a more limited number of syllables than English words; hence, a Vietnamese syllable has many more different meanings than an English word. Only a significant Vietnamese syllable appearing in the immediate context of a target syllable can change the meaning of the target into a total different topic, but this depends much on the distance from the significant syllable model to the target.

In the weighted frequency models, the best weight in English is $(\mathrm{Ln}(T_i)/T_i)$ and Vietnamese $(1/T_i)$. They are different because a Vietnamese sentence is generally longer in syllable length than its corresponding English sentence in word number, for example, the following sentence has 12 syllables " Tôi làm nghiên cứu về mô hình ngôn ngữ cá thể từ " and its corresponding English sentence that means "I do research in individual word language models" only contains 8 words.

7.3 Results for *A Posteriori* Models

We investigated our new approach for calculating *a posteriori* probabilities using five models: the linear interpolation probability model, the exponential probability model, the weighted probability model, the linear interpolation frequency model, and the weighted frequency model. We found that the best performance was provided by the *a posteriori* weighted frequency model, which gave a 44.46% English improvement and 53.34% Vietnamese improvement in perplexity. This is better than the performance of much more computationally intensive methods based on clustering (Iyer & Ostendorf, 1999; Clarkson *et al.*, 1997). The *a posteriori* weighted frequency model's results are displayed in Table 12 and Table 13, and the best results for all different *a posteriori* models are shown in Table 14 and Table 15.

In order to compare to *a priori* models in Table 10, in the *a priori w*eighted frequency model for English, the weights are different, namely $\mathrm{Ln}(T_i)/T_i$ in the *a priori* model and $1/T_i$*$\mathrm{Ln}(T_i)$ in the *a posteriori* model. Besides, the *WM* values, the number of significant word language models or *m* in Equation (14), are quite different, with values of 29 for the *a priori* and 16 for the *a posteriori* models. This can be explained as the concept that people can catch the meaning of a target word more clearly and quickly when they not only hear its previous words but also listen to its following words. Then, the English *a posteriori* model needs to increase weight on significant word models to the global model weight, but it needs to "hear" fewer significant words in the context before it can catch the meaning.

Table 12. The a posteriori weighted frequency model (English WSJ.)

	Whole Sentence Perplexity				**Whole Sentence Improvement**			
Weight	**tri-gram**	**5-gram**	**7-gram**	**9-gram**	**tri-gram**	**5-gram**	**7-gram**	**9-gram**
$T_i*Ln(T_i)$	73.91	56.08	54.61	54.52	1.93%	25.59%	27.53%	27.66%
T_i	73.77	56.00	54.54	54.45	2.11%	25.69%	27.62%	27.75%
$T_i/Ln(T_i)$	73.62	55.92	54.47	54.38	2.31%	25.80%	27.72%	27.85%
$Sqrt(T_i)$	71.88	54.92	53.56	53.48	4.63%	27.13%	28.93%	29.03%
$Ln(T_i)$	67.91	52.52	51.35	51.30	9.88%	30.30%	31.86%	31.93%
$Sqrt(Ln(T_i))$	67.49	52.26	51.10	51.05	10.45%	30.65%	32.19%	32.26%
$Ln(1+Ln(T_i))$	67.34	52.17	51.02	50.97	10.64%	30.77%	32.30%	32.36%
$1/Ln(1+Ln(T_i))$	66.72	51.79	50.66	50.62	11.47%	31.28%	32.78%	32.83%
$Sqrt(1/Ln(T_i))$	66.57	51.69	50.57	50.53	11.67%	31.41%	32.89%	32.95%
$1/Ln(T_i)$	66.07	51.39	50.29	50.25	12.32%	31.81%	33.27%	33.32%
$Sqrt(1/T_i)$	58.31	46.54	45.73	45.72	22.63%	38.24%	39.32%	39.33%
$Ln(T_i)/T_i$	52.00	42.71	42.16	42.19	31.01%	43.32%	44.06%	44.02%
$1/T_i$	51.45	42.44	41.92	41.96	31.73%	43.68%	44.37%	44.32%
$1/T_i*Ln(T_i)$	**51.09**	**42.30**	**41.81**	**41.85**	**32.21%**	**43.87%**	**44.52%**	**44.46%**

Table 13. The a posteriori weighted frequency model (Vietnamese QUB.)

	Whole Sentence Perplexity				**Whole Sentence Improvement**			
Weight	**tri-gram**	**5-gram**	**7-gram**	**9-gram**	**tri-gram**	**5-gram**	**7-gram**	**9-gram**
$T_i*Ln(T_i)$	122.45	98.83	97.16	96.87	0.41%	19.62%	20.98%	21.22%
T_i	122.27	98.68	97.02	96.73	0.56%	19.74%	21.10%	21.33%
$T_i/Ln(T_i)$	122.03	98.49	96.83	96.54	0.75%	19.90%	21.25%	21.48%
$Sqrt(T_i)$	118.30	95.61	94.06	93.79	3.79%	22.25%	23.51%	23.72%
$Ln(T_i)$	107.45	87.82	86.52	86.30	12.61%	28.58%	29.64%	29.82%
$Sqrt(Ln(T_i))$	106.05	86.82	85.55	85.33	13.75%	29.39%	30.42%	30.60%
$Ln(1+Ln(T_i))$	105.64	86.53	85.27	85.05	14.09%	29.63%	30.65%	30.83%
$1/Ln(1+Ln(T_i))$	103.41	84.94	83.72	83.51	15.90%	30.92%	31.91%	32.08%
$Sqrt(1/Ln(T_i))$	102.96	84.62	83.41	83.20	16.27%	31.19%	32.17%	32.34%
$1/Ln(T_i)$	101.24	83.38	82.21	82.00	17.67%	32.19%	33.14%	33.31%
$Sqrt(1/T_i)$	82.11	69.34	68.46	68.31	33.22%	43.61%	44.32%	44.45%
$Ln(T_i)/T_i$	68.88	59.43	58.75	58.63	43.98%	51.67%	52.22%	52.32%
$1/T_i$	67.31	58.35	57.71	57.59	45.26%	52.55%	53.07%	53.16%
$1/T_i*Ln(T_i)$	**66.77**	**58.11**	**57.49**	**57.37**	**45.70%**	**52.74%**	**53.25%**	**53.34%**

Table 14. Improvements of a posteriori models (English WSJ.)

***A Posteriori* Models**	**tri-gram**	**9-gram**	**Best Values**
Global	0.00%	26.77%	
Linear Interpolation Probability	30.99%	44.20%	$\lambda=0.2$
Exponential Decay Probability	28.66%	42.82%	*Decay=100, Cache=5*
Weighted Probability	30.82%	44.17%	$Sqrt(1/Ln(T_i))$
Linear Interpolation Frequency	25.17%	40.66%	$\lambda=0.001$
Weighted Frequency	**32.21%**	**44.46%**	***WM=16, $1/T_i$*Ln(T_i)***

Similar to *a posteriori* Vietnamese models in Table 15, the weighted frequency model has *WM*=34 and a weight of $1/(T_i*Ln(T_i))$, while the *a priori* ones are much larger, *WM*=67 and weights $1/T_i$.

Table 15. Improvements of a posteriori models (Vietnamese QUB.)

***A Posteriori* Models**	**tri-gram**	**9-gram**	**Best Values**
Global	0.00%	20.88%	
Linear Interpolation Probability	37.44%	47.63%	$\lambda=0.3$
Exponential Decay Probability	40.31%	49.28%	*Decay=15, Cache=99*
Weighted Probability	37.62%	47.34%	$Ln(T_i)$
Linear Interpolation Frequency	36.21%	47.03%	$\lambda=0.001$
Weighted Frequency	**45.70%**	**53.34%**	***WM=34, $1/T_i$*Ln(T_i)***

Previously, for similar experiments, Sicilia-Garcia *et al.* (2005) reported improvements of 68%-69% for 7-gram models. Nevertheless, that was based on the flawed calculation previously described in the section, *A Posteriori* Method, so the results we present here can be viewed as a true reflection of the performance that can be achieved by these models. Nowadays, with the current condition of PCs, the programming issues of Sicilia-Garcia in training and accessing a language model of 9-grams no longer exist. With a 9-gram phrase length, on the global database of WSJ 4.5 gigabytes, we now complete the *a posteriori* probability linear interpolation model in 30 minutes, while Sicilia-Garcia finished this computer execution for 7-grams in 4 days nonstop. For the *a posteriori* weighted frequency model, we now execute only over 2 days for 9-grams, while Sicilia-Garcia completed 10 full days for 7-grams. (This is without speeding up the algorithms; it is only by upgrading to newer computers.)

The different nature of the Vietnamese syllabic language causes a much smaller proportion of syllables to be significant than English significant words. In a Vietnamese sentence and an English sentence of the same length, there are considerably fewer significant syllables occurring, hence, inconsistent effects are found from our results.

7.4 Speech word-error-rate Results for *A Posteriori* Models

We employ a Vietnamese large vocabulary continuous speech decoder linking to the five Vietnamese *a posteriori* models. Our speech recognition system is syllable-based HMM trained with Vietnamese speech data of 97,850 spoken statements extracted from the Vietnamese QUB text corpus; they are recordings of 326 speech hours by 60 Vietnamese speakers, including 30 men and 30 women from 18 years old to 51 years old. The number of states per HMM is 5, and each state was modeled using a mixture of 16 Gaussians. It is a tone-dependent phoneme model.

7.4.1 Vietnamese VnIPh Lexicon

In March 2009, Ha created the Vietnamese lexicon called Vietnamese International Phonetic Lexicon version 1.0 or VnIPh lexicon (http://hochiminhcityuniversityofindustry-lequanha.schools.officelive.com/VnIPh.aspx).

We employ this lexicon in our speech recognition system; it includes an automatic lexicon generator to create 12,165 Vietnamese syllable entries. We would like to show a few Vietnamese syllables from this lexicon in Table 16.

Table 16. The Vietnamese VnIPh Lexicon (a few entries)

Vietnamese Syllable	Phoneme	Meaning in English
CẢNG	K 3 AA NG	port
MẸ	M 5 EH	mother
NGHĨA	NG 2 IY AH	meaning
TÔI	T 0 OY	I, me
TÙY	T W 1 IY	depend
TÚ	T 4 UH	excellent

In Table 16, phonemes 0, 1, 2, 3, 4, and 5 are the six tones of Vietnamese speech: level, low, high rising glottalized, dipping-rising, high rising, and low glottalized tones. In fact, there are 43 context-independent phonemes: 0, 1, 2, 3, 4, 5, AA, AH, AO, AW, AY, B, CH, D, EH, ER, EY, F, G, HH, IH, IY, K, L, M, N, NG, OW, OY, P, R, S, SH, SIL, T, UH, UW, V, W, WW, WY, Y, and Z.

7.4.2 Vietnamese Speech Tests

We set up our own Vietnamese speech tests that included 21,451 Vietnamese syllables of 3,363 types in 3,834 statements. In Vietnamese speech, the number of syllables is countable because, using the Vietnamese spelling rule, all of the syllables can be obviously formed from the vowels, consonants, and tones. Hence, similar to English words, the number of Vietnamese words, which are combinations of syllables, is uncountable. Nevertheless, all of the possible Vietnamese syllables are fully stored in our lexicon and there are no

out-of-vocabulary syllables in our Vietnamese speech tests, even though there are still unknown syllable combinations or unknown Vietnamese words out of the training corpus.

Although the Vietnamese QUB training corpus is clean, without any external noises, our Vietnamese speech tests are noisy with all insertion/deletion/substitution cases of phonemes. Noisy spoken tests are recorded in realistic environments, around cars, fans, and other people talking; each utterance is recorded in a high, medium, or low noise condition.

We apply an unconstrained phone recognizer, and the observed phone sequences are obtained. Our expected phone sequences are constructed using lexicon concatenation based on the Vietnamese syllable transcriptions. In comparison to the expected sequences, our observed sequences show three experimental conditions: well-matched, medium-matched, and highly-mismatched. In our tests, we have three common types of phone errors, which are 16.57% insertion, 18.24% deletion, and 16.21% substitution.

7.4.3 Vietnamese Language Models

Our Vietnamese recognition baseline system uses a Katz back-off trigram language model; hence, we can calculate the word error rate (WER) as follows in Table 17.

All of our Katz probabilities of Vietnamese phrases were stored in a hashed file inside the decoder. In order to link our individual word models into this decoder, we first calculated the combined probabilities of all *n*-grams from unigrams and bigrams up to 9-grams via the five models in Equations (6), (7), (8), (12), and (14). This took us ten days; then, we created five hashed files storing these Vietnamese probabilities with the same structure of the existing Katz hashed file.

In the next step, we replaced the Katz file by each of our five files or our individual word language models. In this way, when speaking, the Vietnamese speech decoder is naturally able to execute on-line. Our Vietnamese word-error-rates are done by tests with direct microphone input from six held-out Vietnamese native speakers, four men and two women, ranging from 21 years old to 52 years old.

Table 17. Noisy Speech A Posteriori WER (Vietnamese QUB.)

***A Posteriori* Models**	**WER**	**Improvement over Baseline**
Baseline	45.65%	
Linear Interpolation Probability	33.67%	11.98%
Exponential Decay Probability	35.78%	9.87%
Weighted Probability	33.33%	12.32%
Linear Interpolation Frequency	33.44%	12.21%
Weighted Frequency	**28.87%**	**16.78%**

8. Conclusions

We have described the concept of using individual word models to improve language model performance. Individual word language models permit an accurate capture of the domains in which significant words occur, thereby improving the model performance. The results indicate that individual word models offer a promising and simple means of introducing domain information into an *n*-gram language model.

Humans probably hear the sounds of several words spoken before using a form of human language model to make a sensible sentence from the sounds, particularly when there are corruptions. Therefore, the idea of using the *a posteriori* method to define the domain might be more appropriate than the *a priori* method. We believe that the use of multiple word-domains, which need only large amounts of relatively inexpensive disk space, models the domain environment of any piece of written or spoken text more accurately than any other domain method. Our Vietnamese word-error-rate measurement to test this theory by linking to a speech decoder is also very good. For our future work, we will apply *a posteriori* language models in automatic speech recognition for English and other languages.

Acknowledgement

We would like to thank Dr. Sicilia-Garcia, Dr. P. Donnelly, Dr. D. Stewart, Dr. P. Hanna, and Prof. F. J. Smith, OBE MRIA. Our heartfelt acknowledgement is sent for the Reviewers of CLCLP. Finally, I would like to appreciate my wife, Mrs. Thuy Vo, and sister, Ms. Giang Vo, for their encouragement to complete this paper during winter in Calgary.

Reference

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet allocation. *Journal of Machine Learning Research,* 3, 993-022. DOI: 10.1162/jmlr.2003.3.4-5.993.

Clarkson, P. R., & Robinson, A. J. (1997). Language Model Adaptation Using Mixtures and an Exponentially Decaying Cache. *IEEE International Conference on Acoustics, Speech, and Signal Processing,* 2, 799-802. Munich, Germany.

Donnelly, P. (1998). A Domain Based Approach to Natural Language Modelling. *PhD Thesis*. Queen's University Belfast, Northern Ireland.

Donnelly, P. G., Smith, F. J., Sicilia-Garcia, E. I., & Ming, J. (1999). Language Modelling With Hierarchical Domains. *The 6th European Conference on Speech Communication and Technology,* 4, 1575-1578. Budapest, Hungary.

Good, I. J. (1953). The Population Frequencies of Species and the Estimation of Population Parameters. *Biometrika,* 40, 237-254.

Ha, L. Q., Stewart, D. W., Ming, J. & Smith, F. J. (2008). Individual Word Probability Models. *Web Journal of Formal, Computational & Cognitive Linguistics (FCCL),* Issue 10, Russian Association of Artificial Intelligence.

Hofmann, T. (1999). Probabilistic Latent Semantic Indexing. *Proceedings of the Twenty-Second Annual International SIGIR Conference on Research and Development in Information Retrieval (SIGIR-99)*.

Hofmann, T. (1999). The Cluster-Abstraction Model: Unsupervised Learning of Topic Hierarchies from Text Data. *IJCAI*, 682-687.

Hofmann, T., Puzicha, J., & Buhmann, J. M. (1998). Unsupervised Texture Segmentation in a Deterministic Annealing Framework. *IEEE Trans. Pattern Anal. Mach. Intell.* 20, 803-818.

Iyer, R. M., & Ostendorf, M. (1999). Modeling Long Distance Dependence in Language: Topic Mixture Versus Dynamic Cache Models. *IEEE Transactions on Speech and Audio Processing,* 17(1), 30-39.

Jelinek, F., Mercer, R. L., & Bahl, L. R. (1983). A Maximum Likelihood Approach to Continuous Speech Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 5, 179-190.

Katz, S. M. (1987). Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recogniser. *IEEE Transactions on Acoustic Speech and Signal Processing,* 35(3), 400-401.

Kuhn, R., & De Mori, R. (1990). A Cache-Based Natural Language Model for Speech Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 12(6), 570-583.

Lau, R., Rosenfeld, R., & Roukos, S. (1993). Trigger-based Language models: A Maximum entropy approach. *IEEE International Conference on Acoustics, Speech, and Signal Processing,* 2, 45-48. Minneapolis, MN.

O'Boyle, P., Owens, M., & Smith, F. J. (1994). Average *n*-gram Model of Natural Language. *Computer Speech and Language,* 8, 337-349.

Paul, D. B., & Baker, J. M. (1992). The Design for the Wall Street Journal-based CSR corpus. *The Second International Conference on Spoken Language Processing*, 899-902.

Seymore, K., Chen, S., & Rosenfeld, R. (1998). Nonlinear Interpolation of Topic Models for Language Model Adaptation. *The 5th International Conference on Spoken Language Processing,* 6, 2503-2506. Sydney, Australia.

Sicilia-Garcia, E. I. (2002). A Study in Dynamic Language Modelling. *PhD Thesis*. Queen's University Belfast, Northern Ireland.

Sicilia-Garcia, E. I., Ming, J., & Smith, F. J. (2001). Triggering Individual Word Domains in *n*-gram Language Models. *The 7th European Conference on Speech Communication and Technology,* 1, 701-704. Aalborg, Denmark.

Sicilia-Garcia, E. I., Ming, J., & Smith, F. J. (2002). Individual Word Language Models and the Frequency Approach. *The 7th International Conference on Spoken Language Processing*, 897-900. Denver, Colorado.

Sicilia-Garcia, E. I., Ming, J., & Smith, F. J. (2005). *A posteriori* multiple word-domain language model. *The 9th European Conference on Speech Communication and Technology (Interspeech-Eurospeech)*, 1285-1288. Lisbon, Portugal.

Computational Linguistics and Chinese Language Processing
Vol. 15, No. 2, June 2010, pp. 127-144

Improving the Template Generation for Chinese Character Error Detection with Confusion Sets

Yong-Zhi Chen*, Shih-Hung Wu*, Ping-che Yang+, and Tsun Ku+

Abstract

In this paper, we propose a system that automatically generates templates for detecting Chinese character errors. We first collect the confusion sets for each high-frequency Chinese character. Error types include pronunciation-related errors and radical-related errors. With the help of the confusion sets, our system generates possible error patterns in context, which will be used as detection templates. Combined with a word segmentation module, our system generates more accurate templates. The experimental results show the precision of performance approaches 95%. Such a system should not only help teachers grade and check student essays, but also effectively help students learn how to write.

Keywords: Template Generation, Template Mining, Chinese Character Error.

1. Introduction

In essays written in Chinese by students, incorrect Chinese characters are quite common. Since incorrect characters are a negative factor in essay scoring, students should avoid such errors in their essays. Our research goal is to build a computer tool that can detect incorrect Chinese characters in student essays and correct them, so that teachers and students can learn faster with help from the computer system.

Compared with the detection of spelling errors in English, the detection of incorrect Chinese characters is much more difficult. In English, a word consists of a series of letters while a meaningful Chinese word usually consists of 2 to 4 Chinese characters. The difficulty lies partly in the fact that there are more than 5,000 high-frequency characters.

In previous works on Chinese character error detection systems (Zhang, Huang, Zhou, &

* Department of Computer Science and Information Engineering, Chaoyang University of Technology
E-mail: {9727602, shwu}@cyut.edu.tw
The author for correspondence is Shih-Hung Wu.
+ Institute for Information Industry
E-mail: {maciaclark, cujing}@iii.org.tw

Pan, 2000) (Ren, Shi, & Zhou, 1994), a confusion set for each character is built and is used to detect the character error with the help of a language model. The confusion set is based on a Chinese input method. The characters that have similar input sequences probably belong to the same confusion set. For example, the Wubizixing input method (Wubi), which is a Chinese character input method primarily for inputting both simplified and traditional Chinese text in a computer, is used in (Zhang, Huang, Zhou, & Pan, 2000). The Wubi method is based on the structure of the characters rather than on the pronunciation. It encodes every character in four keystrokes at the most. Therefore, if one keystroke is changed, another character similar to the correct one will show up. Once a student chooses the similar character instead of the accurate one, a character error is established, and a confusion set is automatically generated by the character error. Another approach is to manually edit the confusion set. *Common Errors in Chinese Writings* gives 1477 common errors (National Languages Committee, 1996). Nevertheless, this amount is not sufficient to build a system. Hung manually compiled 6701 common errors from different sources (Hung & Wu, 2008). These common errors were compiled from essays of junior high school students and were used in Chinese character error detection and correction.

Since the cost of manual compilation is high, Chen *et al*. proposed an automatic method that can collect these common errors from a corpus (Chen, Wu, Lu, & Ku, 2009). The idea is similar to template generation, which builds a question-answer system (Ravichandran & Hovy, 2001) (Sung, Lee, Yen, & Hsu, 2008). The template generation method investigates a large corpus and mines possible question-answer pairs. Templates for Chinese character error detection can be generated and tested by the chi-square test on the basis of a large corpus. In this paper, we will further improve the methods for building confusion sets and automatically generating a template.

According to recent studies(Liu, Tien, Lai, Chuang, & Wu, 2009a; 2009b), character errors in student essays are of four major types: errors in which characters have similar shapes (30.7%), errors in which characters have similar pronunciation (79.9%), errors in which the two previous types are combined (20.9%), and other errors (2.4%). Therefore, an ideal system should be able to deal with these errors, especially those resulting from similar pronunciation and similar character shapes. The confusion set for similar pronunciation is relatively easy to build, whereas the confusion set for similar shapes is more difficult. In addition to the Wubi input method, the Cangjie input method is also used to compile confusion sets (Liu & Lin, 2008).

The paper is organized as follows. In Section 2, we introduce the system design and related works. In Section 3, we describe a new process of template generation. Section 4 describes the experimental procedure and the data. Finally, in Section 5, we give the conclusion and propose our future research.

2. System Design

2.1 Chinese Character Error Detection and Correction System

The system that can detect and correct Chinese character errors works as follows. First, it needs a student to input an essay. The system then reports the errors in the essay and gives suggestions on correction, as shown in Figure 1. Such a system uses templates that can detect whether common errors have occurred. A template consists of a pair of words, a correct one and an error one, such as "辯論會"-"辨論會". For example, if the error template "辨論會" is matched in an essay, our system can conclude that there is an error and make a suggestion on correction to "辯論會".

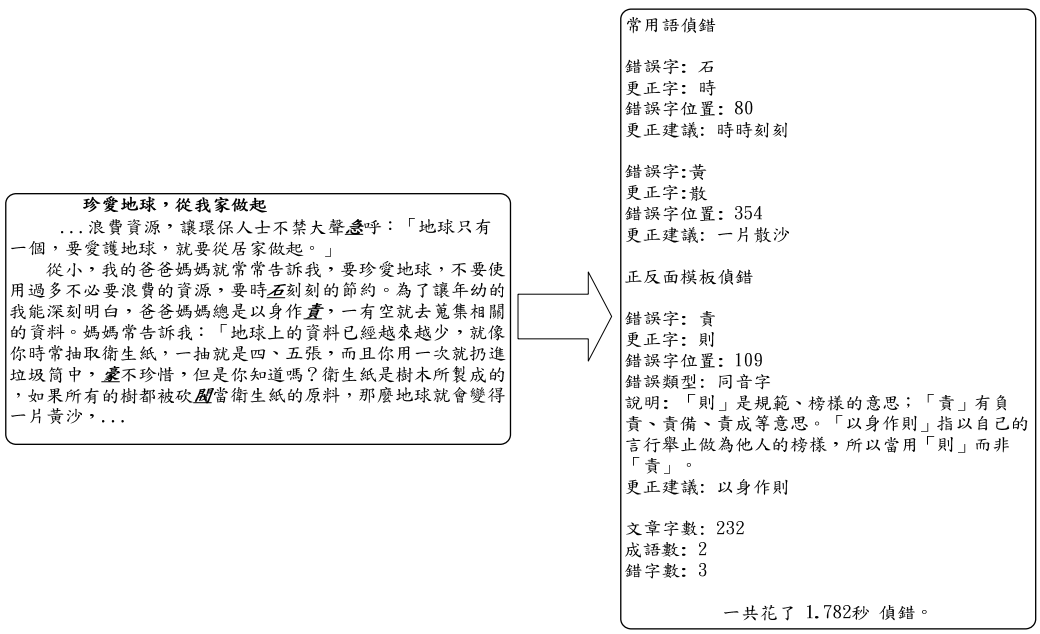


Figure 1. System function of Chinese character error detection in an essay

In previous works, these templates were compiled manually (Liu, Tien, Lai, Chuang, & Wu, 2009b). The quality of the manually-edited templates is high. Nevertheless, the method is time-consuming and costs too much manpower. Therefore, an automatic template generation method based on the context of errors was proposed in 2009 (Chen, Wu, Lu, & Ku, 2009), several examples of automatically generated tri-gram and four-gram templates are shown in Figure 2. The automatic template generation method is less costly; however, it does not accommodate conventional vocabulary. The template generation method has a serious drawback. In Figure 2, we find that several templates contain unrecognizable words, such as "辯護律," "視辯論," and "電視辯," which are trigrams of Chinese characters that do not have

any meaning. These templates can be used to detect character errors, but are not suitable for suggesting corrections.

In the following subsections, we will propose a new method to avoid this drawback.

Templates		Templates	
Correct	Error	Correct	Error
會首長	會首常	清潔隊長	清潔隊常
會給予	會給于	交通隊長	交通隊常
辯論會	辨論會	辯護律師	辨護律師
辯護律	辨護律	視辯論會	視辨論會
的辯論	的辨論	政策辯論	政策辨論
視辯論	視辨論	電視辯論	電視辨論
電視辯	電視辨	公開辯論	公開辨論
半世紀	辦世紀	半個世紀	辦個世紀
半以上	辦以上	一年半的	一年辦的
半個小	辦個小	的另一半	的另一辦

Figure 2. The templates for error detection and correction in (Chen, Wu, Lu, & Ku, 2009)

2.2 Confusion Set

The first step in template generation is to replace one character in a word with a character in the corresponding confusion set. For example, by replacing one character in the correct word "芭蕉," we get a wrong word "笆蕉". Such a correct-wrong word pair is used as the template for error detection and correction suggestion.

According to Liu *et al.* (Liu, Tien, Lai, Chuang, & Wu, 2009a; 2009b), the most common error types are characters with similar shapes and characters with similar pronunciation. The percentage of these two types of errors combined is 89.7% of all errors. Therefore, the confusion set should deal with characters with similar pronunciation and shapes.

We first compile all of the characters that have the same pronunciation from a dictionary and make them the elements of a confusion set. For example, "八(ba1)" and "巴(ba1)" have the same pronunciation. Therefore, they belong to the same confusion set. To reduce the size of the confusion set, we treat characters with different tones as belonging to different sets, even though they sound similar. For example, "罷(ba4)" is not in the confusion set of "八(ba1)". We formed 1,351 sets with a total of 15,160 characters, as shown in Figure 3.

In this paper, we use a simple rule to compile characters with similar shapes. In the first book on Chinese characters, known as Shuowen Jiezi (說文解字) (Xu, 2009), in the second

century, radicals (部首) were used to categorize characters. We use the key component of a character, its radical, as the basic shape of the character to find the characters with the same radicals. There are 214 radicals in Chinese, according to the Kangxi Dictionary (康熙字典) (Zhang, 1999). Therefore, we compile 214 confusion sets with a total of 9,752 different characters. Figure 4 shows some examples.

After constructing the confusion sets, our system can find characters with the same pronunciation and characters with similar shapes for any character that is input. For example, given a character "兇," the system can find characters with the same pronunciation "凶兄匈洶恟胸," and characters with similar shapes "兄光兆先兌克兗," as shown in Figure 5. This is a crucial step of our new template generation.

Zhuyin	Pinyin	Characters
ㄅㄚ	ba1	蚆扒八巴仜叭杁芭疤捌笆粑豝鈀吧
ㄅㄚˊ	ba2	鈸茇拔朳跋菝詙軷魃鼥犮
ㄅㄚˇ	ba3	鈀把靶
ㄅㄚˋ	ba4	伯罷霸猈弝爸壩灞把耙
ㄅㄛ	bo1	剝嚗波袚玻柭砵缽啵菠嗟撥嶓蹳鱍岥播襏
ㄅㄛˊ	bo2	爆伯犮襏挀葡柏跛薄泊礨濼鋍帛勃胉挬浡
ㄅㄛˇ	bo3	簸跛蚾
ㄅㄛˋ	bo4	播檗蘗亳擘譒北挀薜簸繴

Figure 3. Examples of characters in confusion sets

Radicals	Characters
一	一丐丁七三下丈上万丌丑丐不丐丙世丕且
丶	丸凡丹主
丿	乂乃久么之尹乍乏乎乒乓乘乖乘
乙	乙九乜也乞乩乩乳乾亂
亅	了予事
二	二于云井互五亓亙些亞亟
亠	亡亢交亦亥亨享京亭亮亳亶亹
人	人仁什仃仆仇仍今介仄仂仉以付仔仕他仗
儿	兀元允充兄光兇兆先兌克兗兕兔兒兗党兜
入	入內全兩

Figure 4. Examples of characters in confusion sets

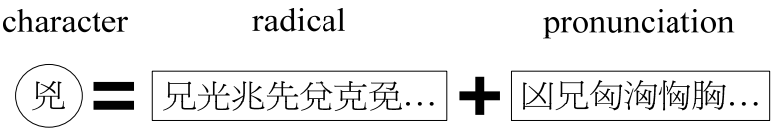


Figure 5. Combination of the two confusion sets for a given character

2.3 Automatic Template Generation

Figure 6 shows the flowchart of our automatic template generation process. The basic assumption is that the corpus might contain more correct words than wrong ones. Therefore, our system first replaces one character in the correct words to form the corresponding wrong words. Then, our system checks the frequency of the words in the corpus. If the replacement creates a word with a relatively high frequency, we do not treat it as a wrong word.

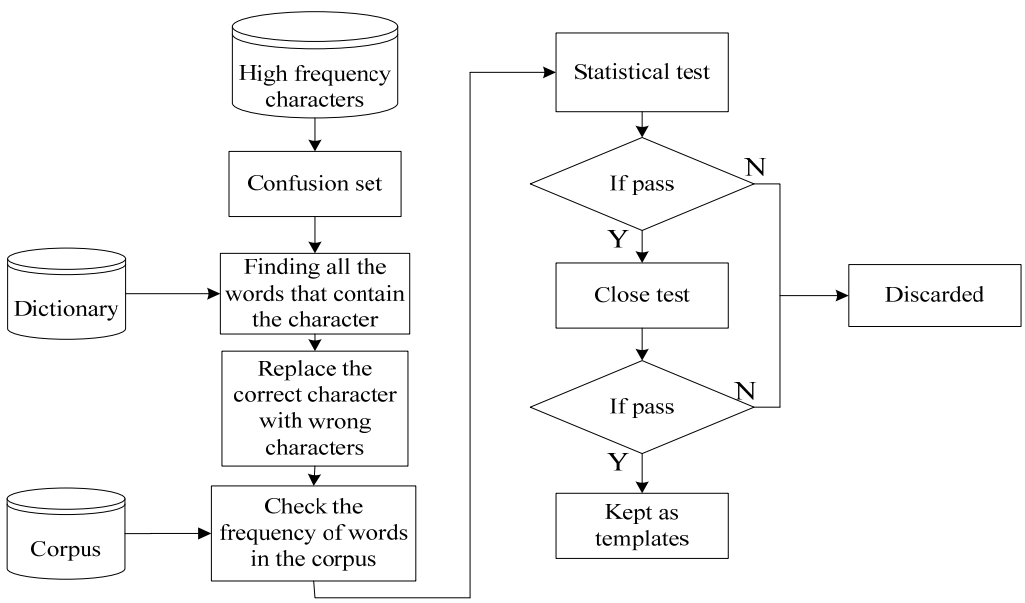


Figure 6. The flowchart of the automatic template generation process

As we mentioned in Section 2.1, the automatically-generated templates might not be suitable for suggesting corrections. To overcome this drawback, we use existing vocabulary, instead of n-gram character sequences, as the candidate for a template. There are 145,608 words in the MOE dictionary (Ministry of Education, 2007). We treat them as the seeds of the templates. In our experiment, we focus on 4,998 high-frequency characters that were compiled on the basis of a 1998 survey (National Languages Committee, 1998).

Our system generates templates by checking each high-frequency character and finding all of the words that contain the character. Then, the system replaces the character in each

word with a character in the corresponding confusion set. The correct-wrong word pair undergoes a simple statistical test. If it passes the test, it will be kept as a template; otherwise, it will be discarded. The statistical test is based on the frequency of each word in the pairs appearing in a large corpus. To prevent the process from generating controversial templates, our system also conducts a close test. The close test checks whether the new template will cause a false alarm on our old test data. The template that generates conflicting templates will also be discarded. The close test threshold is set to 0, which means any template that might cause a false alarm will not be used. A template generation example is shown in Figure 7.

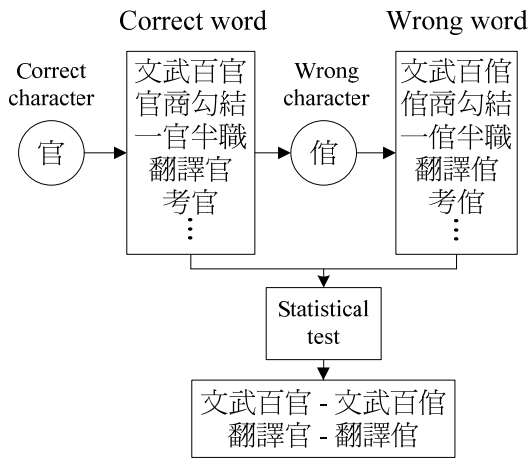


Figure 7. A template generation example, where two templates are generated for an input character "官".

The statistical test in our system is not a rigid test. We tune the threshold of relatively high frequency based on two formulae. One is adopted from the chi-square test, and the other one is from our observation. The first test is a simplified (n=1) chi-square test used in a previous work (Hung & Wu, 2008):

$$X^2 = \frac{(O-E)^2}{E}, \tag{1}$$

where E is the frequency of a correct word and O is the frequency of a wrong word. To avoid further disputation, we assume that E>O in our study. The chi-square test provides a threshold mechanism to decide whether a correct-wrong pair is a proper template or not.

In this study, we suggest the test should be like Equations (2) and (3).

$$\sqrt{Cfreq} > Wfreq \quad , \quad Cfreq > AverageFreq \tag{2}$$

$$Threshold = \frac{\sum_{i=1}^{n} Cvocabulary(i)}{n}, \tag{3}$$

where *Cfeq* is the frequency of the correct word, *Wfeq* is the frequency of the wrong word, and *AverageFreq* is the average of the frequencies of all correct words.

If the frequency of the correct word is higher than the threshold and if the square root of the frequency of the correct word is higher than the frequency of the wrong word, then the pair passes the test.

We have found that the templates that do not pass the test are also the ones that will cause false alarms; for example, the pairs "未來"-"爲來," "已經"-"以經," and "但是"-"但事". When the context is different, these templates do not always give correct detection results and cause false alarms.

2.4 Word Segmentation

As in the examples above, short templates with only two characters could cause false alarms. The reason is that, when we treat words as bi-gram character sequences, many word boundaries may be unclear. For example, as shown in Figure 8, the template "擁有"-"雍有" can be used to detect and correct the first sentence, "一個人可雍有很多快樂", in which one of the word pair appears, but the template "擁有"-"以有" cause a false alarm in the second sentence, "一個人可以有很多快樂". We find that this failure can be avoided by using correct word segmentation. The character "以" should be a part of the previous word "可以". If we have enough confidence in the word segmentation, then the characters in a segmented word should not be candidates for character error detection.

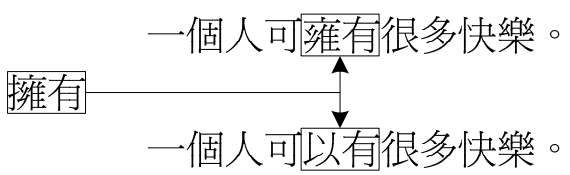


Figure 8. A false alarm in the second sentence for a short template "擁有"-"雍有" and "擁有"-"以有"

We assume that a word segmentation tool can give the correct results for normal input sentences and does not segment sentences with wrong character sequences into words. Figure 9 shows the segmentation results of the two sentences shown in Figure 8. In our experiment, we used the segmentation tool provided by CKIP, Academia Sinica[1]. With the help of this segmentation tool, our system can compile more accurate short templates. Some short templates are shown in Figure 10.

[1] http://ckipsvr.iis.sinica.edu.tw/

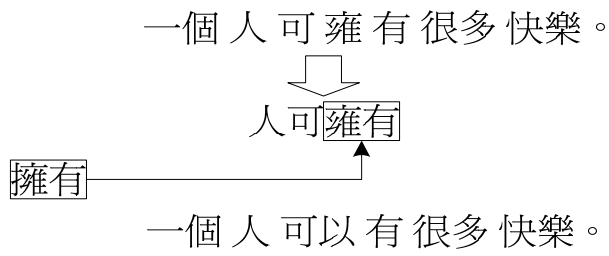


Figure 9. Segmentation tool can help prevent false alarms

Correct templates	Incorrect templates	Correct templates	Incorrect templates	Correct templates	Incorrect templates
衝擊	衝急	絆腳石	伴腳石	逼不得已	逼不得己
檢視	機視	大部分	大不分	情非得已	情非得巳
經濟	經紀	手電筒	手電桶	逼不得已	逼不得巳
循環	循還	不經意	不經易	大勢已去	大勢以去
成績	成積	不願意	不願易	不能自己	不能自以
薪水	新水	董事長	懂事長	迫不得已	迫不得以
賺錢	購錢	三輪車	三軸車	情非得已	情非得以
關鍵	關建	腦震盪	腦振盪	萬不得已	萬不得以
老闆	老版	辦公室	辨公室	逼不得已	逼不得以
雖然	隨然	成績單	成積單	巡弋飛彈	巡曳飛彈

Figure 10. Some short templates generated by our system

3. Experimental Settings, Results and Analysis

3.1 Training Corpus and Student Essays

Our method requires a large corpus to compile templates. Therefore, we used the largest available news corpus as our training set. The corpus is described in Table 1.

Table 1. Corpus statistics

Year	News sources	# of Docs	File size
1998-1999	China Times	38,163	209MB
	China Times Commercial	25,812	
	China Times Express	5,747	
	Central Daily News	27,770	
	China Daily News	34,728	

1998-1999	United Daily News	249,508	320MB
2000-2001	United Daily News	172,421	1.03GB
	United Express	91,958	
	Min Sheng News	168,807	
	Economic Daily News	463,873	

Student essays were collected from one junior high school in Taipei. We used some of the essays for the close test and the rest as the open test, keeping them unseen to the system. The students were 7th or 8th graders. The essays were reviewed by their teachers, and the character errors were highlighted. These 3264 essays were written by hand and were digitized later. See Figure 11 for an example. This is part of our experimental setting that tries to avoid the influence of different input methods. We deleted some symbols and characters that could not be represented by Unicode.

```
<doc>
<class>七年一班</class>
<number>7</number>
<title>藉口</title>
<score>4.5</score>
<essay>
<p>人，有許多夢想，尼采說：「人因夢想而偉大。」雖然是這麼說，不過光「想」是不會有<revise><wrong>認何
</wrong><correct>任何</correct></revise>成果的，古人說：「坐而言，不如起而行」，只是空說說事業<revise><wrong>
終就</wrong><correct>終究</correct></revise>一事無成。</p>
<p>你是否曾找過一些<revise><wrong>冠冕唐荒</wrong><correct>冠冕堂皇</correct></revise>的藉口來遮蓋你的錯誤？
其實這樣是逃避責任的行爲，你不但沒有痛改前非還不斷的替自己的錯誤做辯護、說了謊，表面上看起來是光鮮亮麗，
但「金玉其外，敗絮其中」，赤裸的真實往往是最令人無法接受，真實像一個低等貨，藉口像華美的包裝，一層一層的
將把真實包成了一個人人喜愛的商品，你爲何又要相信藉口的騙局，而不去看背後真實的<revise><wrong>臭陋畫面
</wrong><correct>醜陋畫面</correct></revise>？</p>
<p>人非聖賢，誰能無過？知過能改，善莫大焉，摒除藉口，是一個需要決心、毅力、耐心的工程，我常常聽到一些人想
要出人頭地、事業有成，但他們總是「今天太累了，明天再說。」或「<revise><wrong>我還年青</wrong><correct>我還
年輕</correct></revise>，老了再說」不然就是「再等十年」，許許多多優柔寡斷的回應，不勝枚舉，「及時當勉力，歲
月不待人 」「有花堪折直須折，莫待無花空折枝」時間如水流，沖淡了記憶，帶走了我的童年，也會帶著你的青春。
</p>
<p>燕子去了有再來的時候，<revise><wrong>楊柳估了</wrong><correct>楊柳枯了</correct></revise>有再青的時候，桃
花謝了有再開的時候，時間過了沒有再來的時候，「船到江心補漏遲」，現在努力痛定思痛也許足以彌補以前種種，未
來也許木已成舟無法改變已成定局了！親愛的，時間不等人，有夢想就快去築夢踏實吧！「往者不可諫，來者欲可
追！」</p>
</essay>
</doc>
```

Figure 11. The file format of our test corpus

Table 2 shows the analysis of the student essays. Most of the characters (94%) in use fell into the frequent characters set. Character errors were not very serious for most of the students, with less than 2 character errors per essay.

Table 3 shows our analysis of the character error types. We find that even in written essays, students tend to write characters having the same pronunciation (66~70%). There is also a high percentage of wrong written characters with the same radical (13~16%). Table 4

shows the templates most used for the student essays. These templates are quite common and are too simple for teachers to teach at the 7th and 8th grade levels. A system that can correct these errors may reduce the work of teachers.

Table 2. Analysis of the student essays

	# of Essays	Average score	Average # of characters	Average # of character errors	% of frequent characters
Close test essay	2241	3.62	367.12	1.74	94.23%
Open test essay	1023	3.61	420.02	1.94	94.33%

Table 3. Analysis of the character error types in student essays

	% with the same radical	% with the same pronunciation	% of both	% out of the two main types
Close test essay	13.82%	70.27%	4.92%	20.81%
Open test essay	16.96%	66.31%	2.85%	19.58%

Table 4. The most used templates in the test corpus

Close essay	Correct	已經	變得	自己	景象	一旦	寄託	已經	畢竟	而已	根本
	Wrong	己經	變的	自已	景像	一但	寄托	以經	必竟	而己	跟本
Open essay	Correct	自己	一旦	已經	選擇	煩惱	應該	已經	而已	選擇	後悔
	Wrong	自已	一但	己經	選則	煩腦	因該	以經	而己	撰擇	後侮

3.2 System Evaluation

In this study, we compare the quality of characters manually compiled from books and students with that of automatically generated ones. Since the frequencies of 2-character words, 3-character words, and 4-character words are very different, our system uses different thresholds - 2300, 500, and 100 for 2-character words, 3-character words, and 4-character words, respectively, in the experiment.

The precision and recall are defined as follows:

$$\text{Macro Recall} = \frac{\sum(\frac{dr}{r})}{N} \quad (4) \qquad \text{Macro Precision} = \frac{\sum(\frac{dr}{sd})}{N} \quad (5)$$

$$\text{Micro Recall} = \frac{\sum(dr)}{\sum(r)} \quad (6) \qquad \text{Micro Precision} = \frac{\sum(dr)}{\sum(sd)} \quad (7)$$

where *dr* is the number of correct characters, *r* is the number of character errors, *sd* is the number of character errors that our system detects, and *N* is the number of all of the essays.

Macro Precision and Macro Recall are focused on the performance of correction per essay. This is what real world students might encounter with the system. As Micro Recall and Micro Precision treat the whole data set as one essay, they are suitable for evaluating the average performance of the system. We prefer high precision while maintaining a relatively high recall because we do not want the users to see too many false alarms.

3.3 Experimental Results

We conducted a series of experiments to determine how to improve our system. First, we used confusion sets and the chi-square test to generate templates and compared the performance with the previous work, which did not use confusion sets. Second, we tested whether the square root test is more suitable for our system than the chi-square test. Third, we tested the influence of the segmentation added to our system. We report the best performance of the experimental results by combining the automatically generated templates with the manually edited templates.

3.3.1 The Comparison of Eexperimental Results of Four Automatic Template Generation Settings

Figure 12 shows the experimental results of using the chi-square test in template generation. Setting A used the automatically generated 19,402 templates in the previous work. Setting B used the confusion sets during the process of automatic template generation. The total number of generated templates was 54,253. The performance of the method proposed in this paper is better than the previous work for both precision and recall. Setting C was the automatically generated templates using the confusion set and the square root test. The total number of templates was 50,467. This new setting results in much higher precision. The Macro Precision value is even better than the manually edited Macro Precision value. This result shows that, when we reduce the automatically generated templates with the square root test, we also reduce noise. For Setting D, our system used confusion sets and a word segmentation tool before the square root test, which generated 9,013 templates. We find that the number of templates is reduced while the performance is improved in terms of both Macro Precision and Micro Precision. The trade off is the performance of recall.

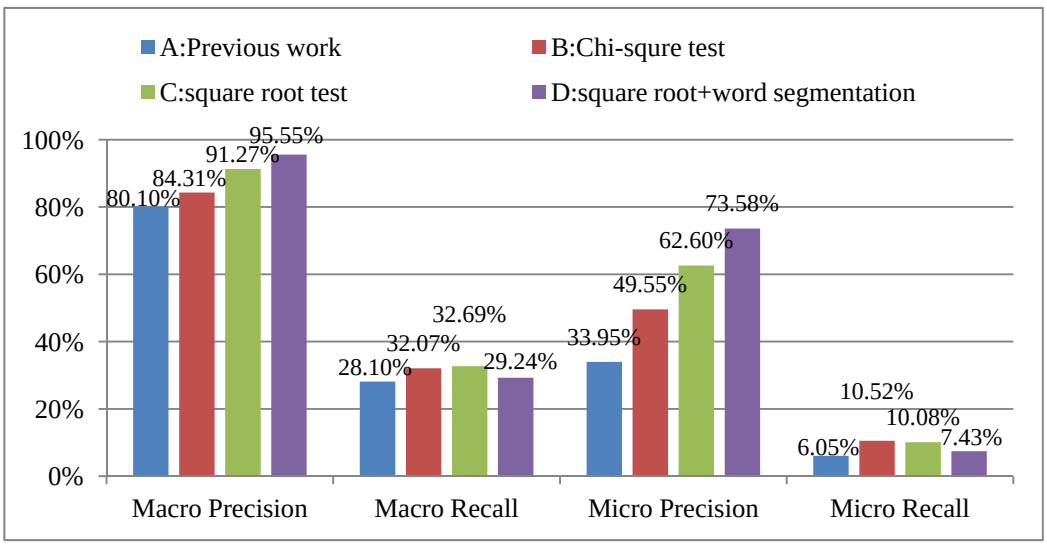


Figure 12. The comparison of experimental results of four automatic template generation settings

3.3.2 Combining Automatically Generated Templates with Manually Edited Templates

Figure 13 shows the comparison of the performance of our system combing automatically generated templates with manually edited templates. Setting E used the 6,701 manually edited templates. Setting F used the combination of Setting E and Setting C, which had a total of 57,167 templates. Setting G used the combination of Setting E and Setting D, totaling 15,713 templates. The performance of the combinations declines a little bit in terms of both Macro Precision and Micro Precision. Nevertheless, there is an increase in both Macro Recall and Micro Recall. Compared with the results in the previous experiment, the combination helps the overall performance. This means that our system can incorporate more templates and attain better performance in the future.

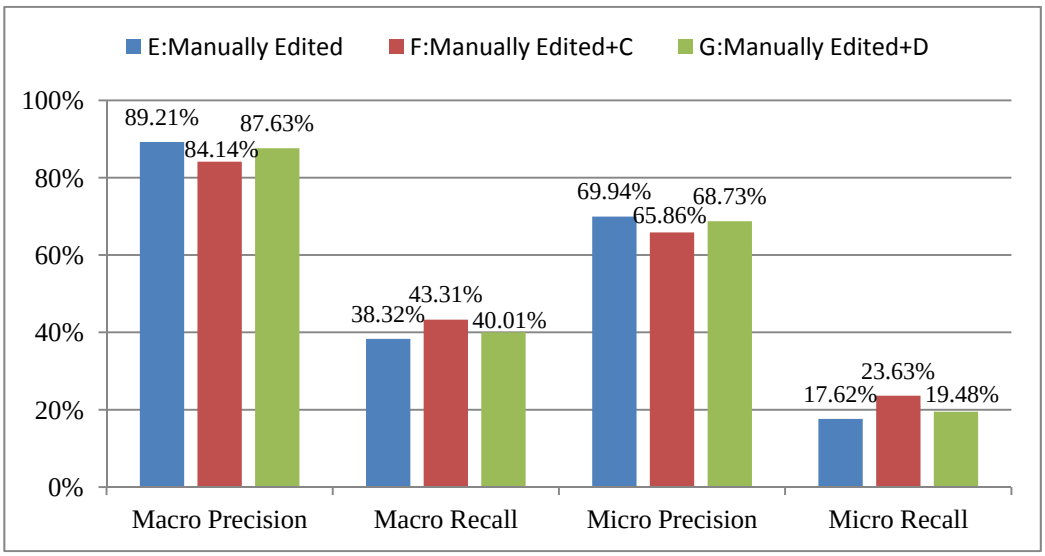


Figure 13. A comparison of experimental results of combining manually edited with automatically generated templates

Based on the analysis of the confusion sets, our system should have a 70% to 80% recall rate because we compile all of the characters with the same pronunciation and some similar characters in the confusion sets. Nevertheless, the recall remains low, even though we are able to control the high-precision performance. Therefore, we will need to conduct further analysis of our system.

3.4 Analysis of the Mistakes in the Experiment

In this subsection, we discuss the 90,135 templates in Setting I of the third experiment, which were generated by using confusion sets, word segmentation, and the square root test. This setting was designed to maintain high precision and to increase recall.

3.4.1 Regarding the Precision

Theoretically, our system can get 100% precision using templates. In practice, however, there are still many exceptions. In Table 5, we list some false alarms in the open tests. According to an online dictionary (Ministry of Education, 2007), some templates that we compiled are interchangeable, such as: “垃圾桶”-“垃圾筒,” “奇蹟” - “奇跡,” “電線桿” - “電線杆,” and “銷聲匿跡” - “消聲匿跡”. This is not consistent with the judgment of some teachers. Some templates are just too short and cannot include the necessary context in order for a correct decision to be made, such as “一再”-“一在”. The necessary context should include more semantic rather than surface syntax. There were some bad templates that our system should

have not generated, such as “放聲大哭”-“放聲大叫,” “不用說”-“不用講,” and “讀書人”-“讀書做,” which can be attributed to the size of the corpus. Nevertheless, no corpus is large enough to be perfect for all applications. We find that these are the major causes of false alarms.

Table 5. Some templates that caused false alarms

Correct word	垃圾桶	奇蹟	電線桿	銷聲匿跡	一再	放聲大哭	不用說	讀書人
Wrong word	垃圾筒	奇跡	電線杆	消聲匿跡	一在	放聲大叫	不用講	讀書做

3.4.2 Regarding the Recall

We treated the errors that the teachers provided from the student essays as templates and compared them to the automatically generated templates, as shown in Table 6. The first column shows the percentage of “not in the automatically generated template”. The second column shows the percentage of an error occurring in a word that is not in the dictionary. The third column shows the percentage of an error occurring in a word that is not in the corpus. The last column shows the percentage of an error occurring in a word that is neither in the dictionary nor in the corpus.

We find that most student errors were not mined from the news corpus, although our system has mined many useful error templates. From the union set of those not in a dictionary and not in a corpus, we find that 53.17% of the necessary templates in the close test set cannot be generated by our system, while 32.97% of the necessary templates in the open test cannot be generated by our system. This is a mismatch of the corpus and student essays. The assumption of our system is that the corpus contains the correct and wrong usages. Nevertheless, since news reporters and junior high school students make character errors for different words, we need to have a more suitable corpus to improve our system. If we have a more contemporary dictionary that includes the words in Table 7, our system can perform better.

Table 6. Comparison of real world errors to system generated templates

	Not matching template	Not in dictionary	Not in corpus	Neither in dictionary nor in corpus
Close test essay	91.53%	37.73%	35.64%	20.20%
Open test essay	93.15%	16.27%	23.94%	7.24%

Table 7. New words not in dictionary

佈告欄	蒸飯機	值日生	作業本	辦派對	睡午覺	全班齊心	勤加練習	羞恥心	無厘頭
重拾信心	莽莽撞撞	淘汰	漆彈場	偶像劇	積陰德	融入團體	芬多精	燒炭	拉筋

4. Conclusion and future works

Based on the confusion sets of Chinese characters, word segmentation, and the square root test, our system can generate a large number of templates from a corpus. These templates can detect and correct Chinese character errors in essays. The templates are more readable and have better performance in both precision and recall performance compared to that of previous system.

To improve the system, we will work in two areas. In the knowledge part, we will enlarge the confusion sets to include more seeds for template generation. We will compile a more suitable corpus for detection and correction of errors in student essays. For the dictionary, we will collect more contemporary terms via the Internet, such as from Wikipedia and Wikitionary. For the language model part, we will use the student essays that we collected in this study to generate an error model, and use that error model to help determine character errors.

Acknowledgement

This study was conducted under the "Project Digital Convergence Service Open Platform" of the Institute for Information Industry, which is subsidized by the Ministry of Economic Affairs of the Republic of China.

Reference

Chen, Y.-Z., Wu, S.-H., Lu, C.-C., & Ku, T. (2009). Automatic Template Generation for Chinese Essay Spelling Error Detecting System. *The 13th Global Chinese Conference on Computer in Education* , 402-408.

Hung, T.-H., & Wu, S.-H. (2008). Chinese Essay Error Detection and Suggestion System. *Taiwan E-Learning Forum* .

Liu, C.-L., & Lin, J.-H. (2008). Using structural information for identifying similar Chinese characters. *Proceedings of the Forty Sixth Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL'08)* , 93-96.

Liu, C.-L., Tien, K.-W., Lai, M.-H., Chuang, Y.-H., & Wu, S.-H. (2009). Capturing errors in written Chinese words. *Proceedings of the Seventh Workshop on Asian Language Resources (ALR7), the Forty Seventh Annual Meeting of the Association for Computational Linguistics (ACL'09)* , 25-28.

Liu, C.-L., Tien, K.-W., Lai, M.-H., Chuang, Y.-H., & Wu, S.-H. (2009). Phonological and logographic influences on errors in written Chinese words. *Proceedings of the Seventh Workshop on Asian Language Resources (ALR7), the Forty Seventh Annual Meeting of the Association for Computational Linguistics (ACL'09)* , 84-91.

Ministry of Education. (2007). *MOE Chinese Dictionary(教育部重編國語辭典修訂本)*. Taiwan: Ministry of Education.

National Languages Committee. (1996). *Common Errors in Chinese Writings (常用國字辨似).* Taiwan: Ministry of Education.

National Languages Committee. (1998). *The Investigation on Common Used Word(八十七年常用語詞調查報告書).* Taiwan: Ministry of Education.

Ravichandran, D., & Hovy, E. (2001). Learning surface text patterns for a Question Answering system. *in Proceedings of the 40th Annual Meeting on Association for Computational Linguistics* , 41-47.

Ren, F., Shi, H., & Zhou, Q. (1994). A hybrid approach to automatic Chinese text checking and error correction. *In Proceedings of the ARPA Work shop on Human Language Technology* , 76-81.

Sung, C.-L., Lee, C.-W., Yen, H.-C., & Hsu, W.-L. (2008). An Alignment-based Surface Pattern for a Question Answering System. *the IEEE International Conference on Information Reuse and Integration* , 172-177.

Xu, S. (2009). *Shuowen Jiezi(說文解字).* Volumes publishing company(萬卷出版公司).

Zhang, Y.-s. (1999). *Kangxi Dictionary(康熙字典).* Chung Hwa Book company(中華書局).

Zhang, L., Huang, C., Zhou, M., & Pan, H. (2000). Automatic detecting/correcting errors in Chinese text by an approximate word-matching algorithm. *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics,* 248-254.

Computational Linguistics and Chinese Language Processing
Vol. 15, No. 2, June 2010, pp. 145-160

以最佳化及機率分佈標記形聲字聲符之研究

Annotating Phonetic Component of Chinese Characters Using Constrained Optimization and Pronunciation Distribution

張嘉惠*、林書彥*、李淑瑩*、蔡孟峰*、李淑萍+、廖湘美+、孫致文+、黃鍔#

Chia-Hui Chang, Shu-Yen Lin, Shu-Ying Li , Meng-Feng Tsai,

Shu-Ping Li, Hsiang-Mei Liao, Chih-Wen Sun, and Norden E. Huang

摘要

一般說來，漢字乃圖形文字，無法像英文等拼音文字一樣，一旦學會拼音方法，即有基本的閱讀能力。相對的，漢字讀寫的學習進展則相當緩慢，而且必須搭配注音符號或是其他拼音方法，才可知道每個漢字的發音。事實上漢字中有八成的字是形聲字，形聲字不僅可由形旁表意，又可以聲符表音，因此即使沒見過的字也可以由偏旁推論其音及義。不過主要的困難在於聲旁未必一定同音，可能是相近的發音，之間的演變規則尚未有人探究過，例如：泡、抱、飽三個字同樣與『包』的發音相近，然而發音如何由『包』的發音轉變成其他三個字的發音，則仍待研究。本論文首先嘗試以自動化方式判定漢字聲符，做爲研究形聲字發聲規則的第一步。實驗顯示，我們所提的兩種方式，發音相似度比較法在 9593 個形聲字中的判定聲符準確率爲 90.7%，而構件發聲分佈比較法則

*國立中央大學資訊工程所，台灣桃園縣中壢市中大路 300 號
Dept. of Computer Science and Information Engineering, National Central University, Taiwan
E-mail: chia@csie.ncu.edu.tw
The author for correspondence is Chia-Hui Chang.
+國立中央大學中文系
Dept. of Chinese Literature, National Central University
#國立中央大學數據中心
Research Center for Adaptive Data Analysis, National Central University

可達到 98.1%的準確率，可以加速形聲字聲符標記所需的大量人力工作與時間。

關鍵字: 形聲字、聲符、發音相似度、最佳化、機率分佈、KL divergence

Abstract

Generally speaking, Chinese characters are graphic characters that do not allow immediate pronunciation unless they are accompanied with Mandarin phonetic symbols (zhuyin) or other pinyin methods (e.g. romanization system). In fact, about 80 to 90 percents of Chinese characters are pictophonetic characters which are composed of a phonetic component and a semantic component. Therefore, even if one had not seen the character before, one can make a logical guess at the character's pronunciation and meaning from its phonetic and semantic symbols. In order to analyze such relations, we start by analyzing the characteristics of phonetic components. We found two interesting features that could automatically identify the phonectic components of Chinese characters. One is pronunciation similarity, the other is pronunciation distribution. Experiments show that these two methods have high accuracy (90.8% and 98.1% for 9593 pictophonetic characters) in predicting the phonetic components of pictophonetic characters. These methods can save a lot of time and effort during the annotation of phonetic symbols in the early stage.

Keywords: Picto-phonetic Compounds, Phonetic Component, Pronunciation Similarity, Pronunciation Distribution, Optimization.

1. 簡介

漢語字形及音讀的繁複，向爲初學者及外籍人士所苦，即使會說華語的海外華人對於漢字的認識也可能相當有限。最主要的原因在於漢字乃圖形文字(pictograph system)，無法像英文等拼音文字(alphabet system)一樣，一旦學會拼音方法(phonetic representation)， 即有基本的閱讀能力。相對的，漢字讀寫的學習進展則相當緩慢，而且必須搭配注音符號(Chinese phonetic symbols)或是其他拼音方法，才可知道每個漢字的發音。這樣的限制，對於漢字的學習相當不利，這也是爲什麼二十世紀初期許多中國革命家意欲將漢字拉丁化的主要原因。

漢字的構成包含象形、指事、會意、形聲、轉注、假借(總稱六書(許慎，1988))，其中象形、指事是「造字法」，會意、形聲是「組字法」，轉注、假借是「用字法」。事實上形聲字所占的比例相當高，約佔八、九成。形聲字不僅可由形旁表意，又可以聲符表音，因此即使沒見過的字也可以由偏旁推論其音及義，所謂『有邊讀邊沒邊讀中間』即是此意。不過主要的困難在於聲旁僅代表相近的發音，之間的演變規則尚未有人探究

過，例如：泡、抱、飽三個字同樣與『包』的發音相近，然而發音如何由『包』的發音轉變成其他三個字的發音，則仍待研究。

爲了解漢字中形聲字與其聲符之間發音規則的轉變，我們必須先知道每一個形聲字的聲符。由於形聲字與聲符發音相近，因此我們憶測可用形聲字與其構件之間的發音相似度做爲聲符預測方法，因此我們一方面尋求聲母、韻母之間發音相似度，一方面也建立一個形聲字源標記系統，由中央大學中文所四位研究生與三位教授參與人工標記漢字構形資料庫中 14706 有注音標示的漢字是否爲形聲字以及其聲符構件，做爲預測方法的驗證。另外爲了提升經由發音相似度比較法判斷聲符之準確率，我們也採用限制性最佳化技術，自動求得新的發音相似度分數。

不過發音相似度比較法僅參考單一漢字的資訊，事實上做爲聲符構件的漢字，其衍生字的發聲分佈比非聲符構件的漢字發聲分佈更爲集中。因此我們進一步提出構件發聲分佈比較法，經由計算每個構件的發聲分佈與所有漢字的發聲分佈的 KL 值，做爲此構件做爲聲符的強度代表。實驗結果顯示，發音相似度比較法在 9593 個已標示的形聲字中判定聲符準確率爲 90.8%，而構件發聲分佈比較法則可達到 98.1%的準確率，顯示兩種方法做爲聲符判斷問題的可行性。

雖然形聲字的聲符預測並非計畫最終目的，而且在聲符標注完成後，預測聲符的需求即消失不在，但是透過發音相似度最佳化方法所得的聲母，韻母相似度參數或許有助於未來漢字字音處理的研究，同時部件發音強度也可做爲漢字學習順序之參考，仍有相當的重要性。

2. 相關研究

漢字的使用從殷商的甲骨文算起已達 3,400 年之久，由於結構複雜，因此文字學在中國特別發達。文字學的研究包括文字起源、發展、性質、體系，文字的形、音、義的關係，正字法以及個別文字演變的情況等等議題。爲有系統的研究，中央研究院資訊科學研究所文獻處理實驗室從 1993 年開始，陸續建構古今文字的源流演變、字形結構及異體字表，做爲記錄漢字形體知識的資料庫，也就是漢字構形資料庫(莊德明、謝清俊，2005)。漢字構形資料庫不僅銜接古今文字以反映字形源流演進，也記錄了不同歷史時期的文字結構。另外也由於開發漢字部件檢字系統，得以解決缺字問題。

然而過去的研究著重在字形知識的整理，尚未涉及字音與字義的處理；因此近年來開始文字學入口網站建置計畫(莊德明、謝清俊，2005；莊德明、鄧賢瑛，2008)。一如其文所述：

“漢字構形資料庫目前只著重在字形知識的整理，尚未涉及字音與字義；建立一個形、音、義俱備的漢字知識庫，仍是我們長遠的目標”，因此本計畫“漢語系統音源脈絡之分析”的目的即是以挑戰漢字的發音規則知識庫爲出發，除了了解漢字發音規則外，也希望藉由此項研究找出一套形聲字發音轉換規則，讓華語學習可以在聲符與規則的輔助下，順利讀出字的發音。爲達成此目的，第一步我們必須了解每個漢字是否爲形聲字，

以及了解形聲字聲符的部件，進而解析聲符與最終發音之間轉換的規則。因此我們首先設計一個“形聲字聲符標記系統”，由中文系研究生與教授的協助，進行形聲字與其聲符的標記。不過由於此過程需耗費大量時間與人力投入（在 2009/11/10 至 2010/11/23 期間內共有 9593 字由至少三位研究生標記相同聲符），因此是否存在自動判定形聲字聲符的方法，變成本計畫第一個挑戰。

3. 發音相似度比較公式表

一般說來，聲符構件通常與原字的發音相似度高於非聲符構件與原字的發音相似度，舉例來說，“詁”字與其聲符構件”古”發音相同，而與其非聲符構件“言”發音較不相似，又如“校”字與其聲符構件”交”發音相近，而與其非聲符構件“木”發音較不相似。因此發音相似度可以做爲我們判定一個形聲字聲符的重要依據。爲此我們必須有一套漢字發音相似度的計算方法。

每一個單獨的漢字雖爲單音節發音，但是就聲韻學上來看可分爲聲母、韻母與調性三類。聲母是使用在韻母前面的輔音，隨著發音部位與發音方法而有所不同，如表一所示；而韻母則是一個音節中的元音（母音），也是押韻的主要部份；再者由於漢語本身是具備聲調系統的語言，因此我們在計算一個形聲字與其構件的發音相似度時即可以聲母、韻母、及聲調的相似度做爲判斷發音相似度的依據。

表一、聲母的發音部位與發音方法

發音部位		上阻	上唇	上齒	齒背	上齒齦	前硬顎		軟顎
發音方法		下阻	下唇		舌尖		舌尖後	舌面前	舌面後
狀態	聲帶	簡稱 氣流	雙唇	唇齒	舌尖前	舌尖中	舌尖後	舌面前	舌根
塞	清	不送氣	ㄅb [p]			ㄉd [t]			ㄍg [k]
		送氣	ㄆp [pʰ]			ㄊt [tʰ]			ㄎk [kʰ]
塞擦	清	不送氣			ㄗz [ts]		ㄓzh [tʂ]	ㄐj [tɕ]	
		送氣			ㄘc [tsʰ]		ㄔch [tʂʰ]	ㄑq [tɕʰ]	
擦	清			ㄈf [f]	ㄙs [s]		ㄕsh [ʂ]	ㄒx [ɕ]	ㄏh [x]
	濁						ㄖr [ʐ]		
鼻	濁		ㄇm [m]			ㄋn [n]			
邊	濁					ㄌl [l]			

3.1 人工制定聲母和韻母發音相似度

如表一所示，聲母依發音部位的不同，可分爲雙唇、唇齒、舌尖前、舌尖中、舌尖後、舌面前、舌根七種音；若依發音方法的不同，則可分爲塞音、塞擦音、擦音、鼻音、邊音等五類，加以聲帶的清濁不同，我們制訂如下之聲母與聲母之間發音相似度：

相同的聲母，相似度訂爲 ub=1。

國際拼音相同，但是差一個上標號，相似度訂爲 a=0.9。

發音部位相同（同列），相似度訂爲 b=0.8。

發音方法相同（同行），相似度訂爲 c=0.5。

其他情形（如不同行不同列或是沒有聲母的情形），相似度訂爲 0.1。

同時我們也尋求語言學專家的協助，針對此版本的制訂規則提供意見，專家們對聲母部分提出以下兩項的建議：

ㄈ與ㄅㄆㄇ的關係合併爲同一行，隸屬唇音一大類。相似度訂爲 b。

ㄗㄘㄙ、ㄓㄔㄕ與ㄐㄑㄒ三大類(不包括ㄖ)，除了同屬塞擦音、擦音外，在發音上有許多人有彼此相混的現象，所以相似度定爲 d=0.7。

表二、韻母種類

種類	注音符號
單韻母	(ㅍ)、ㄧ(yi)、ㄨ(wu)、ㄩ(yu)、ㄚ(a)、ㄛ(o)、ㄜ(e)、ㄝ( )
複韻母	ㄞ(ai)、ㄟ(ei)、ㄠ(ao)、ㄡ(ou)
聲隨韻母	ㄢ(an)、ㄣ(en)、ㄤ(ang)、ㄥ(eng)
捲舌韻母	ㄦ(er)
結合韻母	ㄧㄚ(ya)、ㄧㄛ(yo)、ㄧㄝ(ye)、ㄧㄞ(yai)、ㄧㄠ(yau)、ㄧㄡ(you)、ㄧㄢ(yan)、ㄧㄣ(yin)、ㄧㄤ(yang)、ㄧㄥ(ying)
	ㄨㄚ(wa)、ㄨㄛ(wo)、ㄨㄞ(wai)、ㄨㄟ(wei)、ㄨㄢ(wan)、ㄨㄣ(wen)、ㄨㄤ(wang)、ㄨㄥ(weng)
	ㄩㄝ(yue)、ㄩㄢ(yuan)、ㄩㄣ(yun)、ㄩㄥ(yong)

接著我們制訂韻母與韻母的發音相似度。雖然每個韻母在注音符號表中只是單一個符號，但是我們可以進一步將韻母分爲單韻母、複韻母、聲隨韻母、捲舌韻母與結合韻母五種。其中複韻母包括兩個母音，聲隨韻母包含韻母後的輔音，而結合韻母則包含兩個韻母。依據以上分類，我們制訂韻母發音相似度如下：

韻母相同則相似度訂爲 1。

若兩個韻母同爲結合韻母，若兩者有共同尾音則設相似度爲 x=0.8，若兩者有共同第一個音則設相似度爲 y=0.5，否則設相似度爲 lb=0.1。例如：ㄧㄚ和ㄨㄚ有共同尾音ㄚ因此相似度爲 x、ㄨㄤ和ㄨㄢ有相同第一個韻母因此相似度爲 y；ㄧㄞ和ㄨㄟ因無共同部分，設相似度爲 lb。

若兩個韻母均非結合韻母（也就是屬於單韻母、複韻母、聲隨韻母或捲舌韻母），則相似度以國際拼音決定。若拼音出現相同字母則設相似度爲 z=0.5，否則設相似度爲 lb。例如：ㄧ(i)和ㄞ(ai)有相同部分(i)、ㄠ(au)和ㄡ(ou)間有相同部分(u)皆設相似度爲 z。

若一則爲結合韻母，一則非結合韻母，則給分方式如下所示：若單韻母出現在結合韻母中的後面位置設相似度爲 x=0.8，若出現在前面則設相似度爲 y=0.5，否則相似度爲 lb。例如:ㄧ出現在ㄧㄚ前面位置，則設相似度爲 y=0.5；而ㄞ出現在ㄧㄞ後面位置，設相似度則爲 x=0.8。

其他情形，相似度訂爲 lb。

根據上述規則所制訂的聲母與韻母相似度表分別列於附錄 A 及 B。給定以上相似度，我們定義兩個漢字發音相似度爲其聲母相似度、韻母相似度的總和：

$$\text{Similarity}(c1,c2) = \text{Initial}(c1,c2) + \text{Vowel}(c1,c2) \quad (1)$$

因此給定一個形聲字，我們依據漢字構詞資料庫所拆解成的二至三個構件，分別計算這些構件與原本漢字的發音相似度，查閱聲母與韻母的相似度比較公式表，求算聲母與韻母的總和，取相似度大者構件，做爲聲符的預測。因此形聲字 w 的聲符即可選取與 w 發聲最爲相似的構件 c。

$$PC(w) = \arg\max_{c \in w} Similarity(w,c) \quad (2)$$

舉例來說，漢字「校」(ㄒㄧㄠ 4)的構件爲「木」(ㄇㄨ 4)和「交」(ㄐㄧㄠ 1)，採用相似度比較公式表求算「木」和「 校 」的分數，其中聲母不同行不同列，相似度爲 0.1，而韻母無共同音因此相似度則爲 lb，故相似度總和爲 0.1+lb =0.2；同理「交」和「校」的聲母發音部位相同，相似度爲 b，且韻母同爲結合韻母，相似度爲 1，故相似度總和爲 b+1 =1.8，因此系統判定「交」爲「校」的聲符。若各構件的總和皆相同，則加入調性進行校正，選擇與原字調性相同的構件做爲預測聲符。舉例而言，漢字「祖」(ㄗㄨ 3)的構件爲「示」(ㄕ 4)和「且」(ㄑㄧㄝ 3)，採用相似度比較公式表求算「示」和「祖」的相似度分數爲 d+0.1=0.8，「且」和「祖」的總和爲 d+lb=0.8，兩者相似度分數相同，因此我們再加上調性判別，預測與「祖」聲調相同的「且」爲聲符 。

3.2 發音相似度最佳化

由於前述發音相似度比較公式表是由人工制訂，這些值是否能有效的做爲聲符預測的參數，還是有更佳的值可以推測形聲字聲符？另外，聲母、韻母及聲調三者所占的比重爲何？則是本節所要探討的問題。我們嘗試採用限制型最佳化方法計算聲母和韻母之發音相似度。假設一組已知聲符的形聲字 T，依照發音相似度比較公式，我們可以爲每一個形聲字 w∈T 列出 w 的聲符構件與原字發音相似度必須大於非聲符構件與原字發音相似度的限制條件。以前例漢字「校」來說，其構件爲「木」和「交」，而其已知聲符爲「交」，因此 Similarity(木,校)≤Similarity(交,校)，也就是 0.1+lb ≤ b+1。

在我們的問題中，可以將聲母發音相似度參數 ub, a, b, c, d 以及韻母發音相似度參數 x, y, z, lb， 拿來做爲最佳化問題中的變數。由於當限制條件多於變數個數時，系統可能無解，因此我們對每個不等式的聲符部份加上一個額外的變數 $\varepsilon_i \geq 0$，也就是 d+0.1≤ b+1+ ε_i，再以 $\sum_i \varepsilon_i^p$ 做爲最小化的目標函數，確保聲符與原字的發音相似度大於非聲符構件

與原字的相似度。舉例而言，若是聲符與原字的相似度小於非聲符構件與原字的相似度，則 ε_i 必須大於 0 才足以讓條件成立，反之若聲符與原字的相似度已大於非聲符構件與原字的相似度，則 ε_i 在最小化的目標下自然會是 0。因此若有 m 個已知聲符的漢字，則可化爲以下最佳化問題：

$$\min \sum_i \varepsilon_i^p \text{ s.t. } \begin{cases} 0.1 + lb <= ub + 1 + \varepsilon_1 \\ 0.1 + y <= ub + 1 + \varepsilon_2 \\ \text{M} \\ b + y <= a + 1 + \varepsilon_{|T|} \\ a, b, c, d, x, y, z, ub, lb, \varepsilon_i >= 0 \end{cases} \tag{3}$$

其中 p > 0，代表錯誤聲符對系統的處罰程度的不同。在最極端的情形下，我們可以針對 21 個聲母之間的相似度設定 22*21/2 個變數（含聲母空的情形），同理也可對 39 個韻母（含空韻）之間的相似度設定 39*38/2 個變數，再以限制型最佳化的方法來找出對聲符預測最有利的相似度分數。但由於會有將近 1000 個變數，所花的時間相對也比較長。

機率分佈比較法

除了前述兩項發音相似度比較公式表與最佳化分析的方法，我們也從另一個角度觀察漢字的發音，我們發現某些漢字構件有較強的發音強度，常常做爲聲符，而屬於部首的構件，則通常代表字的形意。於是我們假設漢字的發音有可能是由其構件發音強度較高的構件所支配。因此，如何制定構件的發音強度而又不耗費大量的人力是我們的首要目標。我們觀察一些常見的漢字，如表三所示。從中不難發現包含構件「包」的漢字的發音不管在聲母、韻母或聲調的表現一致性較高，而包含構件「火」的漢字則較低，一致性較高的構件我們也可以說它是發音集中在某幾種發音上，反之較爲分散。集中度較高的構件就很有可能支配著包含此構件的漢字發音，也就是構件發音強度較強。

假設 S 代表某些漢字所形成的集合，f(S)、g(S)、h(S)分表示其聲母、韻母及聲調的分佈機率。令 A 表示所有漢字所成的集合，則f(A)、g(A)、h(A)分別表示漢字的聲母、韻母及聲調的分佈機率。同理對於一個漢字構件 b，我們可以找出包含 b 的所有漢字 B，同時求得其聲母、韻母及聲調的分佈機率f(B)、g(B)、h(B)。若是 b 發音集中度較高，則其聲母分佈f(B)與f(A)就會有較大的差異。因此我們採用 Kullback–Leibler divergence 的方法來計算兩個分佈的距離。Kullback–Leibler divergence 的公式如下:

$$KL(\text{P}\|\text{Q}) = \sum_i \text{P(i)} \log \frac{\text{P(i)}}{\text{Q(i)}} \tag{4}$$

因此我們可以計算 KL(f(B) || f(A))做爲構件 b 聲母強度，同理計算 KL(g(B) || g(A)) 做爲 b 韻母強度，以及計算 KL(h(B) || h(A)) 做爲聲調強度。以下我們以調號以下我們以聲母爲例，計算構件「火」及「包」的聲母強度，我們從漢字構詞資料庫中找出所有標示注音的字共|A| = 14598；我們同時統計含有構件「火」的字共有|B|=259，含有構件「包」的字共有|C|=32，其聲母分佈如下:

表三、全部漢字A 及包含構件「火」B 的聲母分佈狀況

	ㄅ	ㄆ	ㄇ	ㄈ	ㄉ	ㄊ	ㄋ	ㄌ	ㄍ	ㄎ	ㄏ
所有漢字 |A|=14958	660	445	564	445	642	630	312	1009	587	380	742
f(A)	0.04	0.03	0.03	0.03	0.04	0.04	0.02	0.06	0.04	0.02	0.05
含構件火 |B|=259	8	5	10	7	6	13	2	18	7	6	25
f(B)	0.03	0.01	0.03	0.02	0.02	0.05	0.007	0.06	0.02	0.02	0.09
含構件包 |C|=32	15	16	0	1	0	0	0	0	0	0	0
f(C)	0.46	0.5	0	0.03	0	0	0	0	0	0	0

	ㄑ	ㄒ	ㄓ	ㄔ	ㄕ	ㄖ	ㄗ	ㄘ	ㄙ	空
所有漢字 |A|=14958	719	931	835	540	567	239	371	294	356	2120
f(A)	0.04	0.06	0.05	0.04	0.03	0.01	0.02	0.02	0.02	0.14
含構件火 |B|=259	8	27	14	11	8	8	8	4	1	45
f(B)	0.03	0.1	0.05	0.04	0.03	0.03	0.03	0.015	0.003	0.17
含構件包 |C|=32	0	0	0	0	0	0	0	0	0	0
f(C)	0	0	0	0	0	0	0	0	0	0

將|A|與|B|正規化得 f(A)與 f(B)兩機率分佈。最後將 f(B)及 f(A)代入 KL-divergence 公式可得構件「火」的聲母強度。同樣方式可計算「包」的聲母強度。

當我們要判斷某個漢字的聲符時，只需要將此漢字的所有構件的聲母、韻母及聲調三種 Kullback–Leibler divergence 的值做加總，那麼擁有最大的加總值的構件便會被我們判定爲此漢字的聲符。這個方法的好處在於方法簡單而且不需訓練過程，後續實驗將可看出這個方法對形聲字聲符判斷相當有效。表四分別列出依聲母、韻母、聲調 KL 值與其構件出現次數|B|的乘積值排序前十名構件。這樣子的排序可以做爲聲符學習的參考順序。

表四、漢字構件發音強度表

字碼	聲母 KL 值	字碼	韻母 KL 值	字碼	聲調 KL 值
分	0.9768	非	1.3864	皇	0.4851
莫	1.3439	分	1.1621	盧	0.5487
非	0.9789	令	1.4116	令	0.2977
令	1.0335	票	1.4535	會	0.4988
元	1.7167	莫	1.439	夷	0.5487
票	1.1263	屯	1.6778	希	0.5458
卑	1.0746	龍	1.1123	余	0.3386
弗	1.4473	皇	1.6968	吉	0.3332
方	0.9731	包	1.3036	肖	0.2747
俞	1.0632	同	1.4166	世	0.4988

4. 實驗

以下實驗探討前二節所提出的三種分析方法：發音相似度比較公式表、發音相似度最佳化與機率分佈比較法，分別在自動判別形聲字之聲符的效能爲何。實驗中使用的測試資料集，是取自漢字構形資料庫中有注音標示的漢字，在 2009/11/10 至 2010/11/23 期間內共有 9593 字爲四位中文系研究生共同標記完成。

4.1 發音相似度比較法

首先我們進行發音相似度比較公式的預測效能。如表五所示，原始發音相似度方法，準確率約 88.67%，其中包含 518 筆無法判別的字；加入聲調的判別，無法判別的字減少至 291，準確率約 90.21%。採用專家建議的修正版之發音相似度比較公式表來測試，準確率提高至 89.39%，再加入聲調後則爲九成零七的準確率。顯示以發音相似度比較公式進行判別聲符，有一定的效果。也因爲考慮聲調可以提升一個多百分點，因此我們將調的相似度直接納入發音相似度的計算，將計算式(1)改爲如下計算式：

$$\text{Similarity}(c1,c2) = \text{Initial}(c1,c2)+\text{Vowel}(c1,c2)+\text{Tone}(c1,c2) \quad (5)$$

其中聲調相似度依兩個發音的聲調是否相同，給予δ及 0 的相似度。參數δ的值則由分析每一個字的 pcdiff 的分佈圖而得。方法如下：對於每一個字 w，我們先用方程式(1)計算 w 與其聲符構件 pc_w 的發音相似度，以及 w 與其非聲符構件的發音相似度（若有多個非

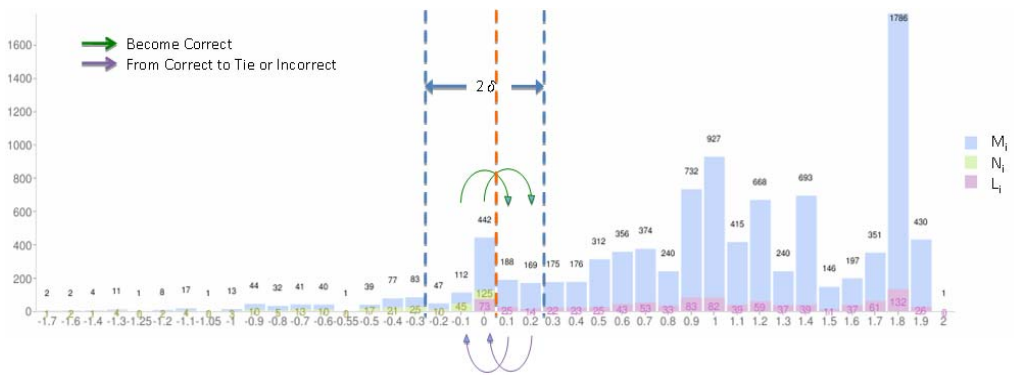
聲符構件則取發音相似度值較大者），並定義 pcdiff(w)爲兩者的差：

$$pcdiff(w) = Similarity(w, pc_w) - \underset{c \in w \setminus pc_w}{similarity(w, c)} \quad (6)$$

表五、發音相似度比較法判別準確率

	正確	錯誤	無法判別	準確率
原始版(no d)	8507	568	518	88.67
原始版＋調	8654	648	291	90.21
修正版(d=0.7)	8576	575	442	89.39
修正版+調	8701	648	244	90.70
整合版(δ=0.2)	8707	618	268	90.86

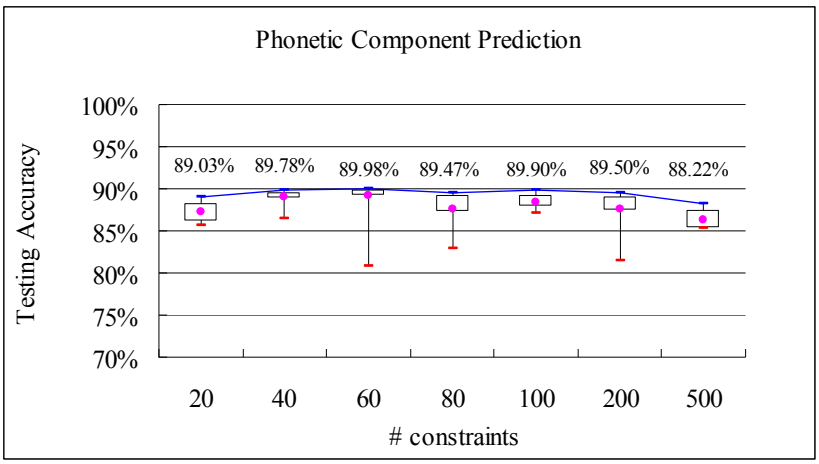
圖一爲所有字的 pcdiff 分佈圖（Histogram）。每一個藍色長條圖 M_i 代表 pcdiff 爲 i（i=…, -0.2, -0.1, 0, 0.1, 0.2, ….）的字的個數（黑色字），同時我們也統計每一藍色長條中有多少個字的聲調與其聲符構件相同但與非聲符構件不同，我們用 N_i 來表示（綠色字）；另外我們也統計有多少個字的聲調與其聲符構件不同但與非聲符構件相同，我們用 L_i 來表示（紫色字）。這兩部份的字集分別代表的是在採用方程式(5)來計算發音相似度時，pcdiff 會增加或是減少的字數。因此如果我們採用方程式(5)來計算發音相似度，將會有新增加 $N_0+N_{-0.1}+\cdots+N_{-\delta+0.1}$ 可被正確預測的字（橘色轉換），但是同時會有 $L_{0.1}+L_{0.2}+\cdots+L_{\delta}$的字會從正確預測轉爲錯誤預測或無法判斷（藍色轉換）。因此若 N_0=125, L_0=73, $N_{-0.1}$=45, $N_{-0.2}$=10, $L_{0.1}$=25, $L_{0.2}$=14（如圖二），當我們設δ=0.2，則我們可新增 125+45=170 個可正確判斷的字，但失去 25+14=39 原可正確判斷的字。整體來說，我們可多預測 170-39=131 個正確的字，而無法判別的字則減少 125+73-10-14＝174 個字(442-174=268)。



圖一、聲符發音相似度與非聲符發音相似度差分佈圖

4.2 發音相似度最佳化

第二部份實驗主要是了解最佳化方法計算出的聲母與韻母相似度參數在判別漢字聲符之可能性。實驗主要目的在了解需要多少訓練資料筆數才能達到大約九成的準確率，以及最佳化方法的學習曲線。由於變數不多，理論上我們所需要的訓練資料筆數並不需要太多，但是若是取太少訓練資料，則預測準確的效果差異則會變大。我們取 p=1 做爲目標函數，隨機抽取 1000 筆資料做爲訓練資料，剩餘 8593 筆資料做爲測試資料。每次從訓練資料中隨機抽取 m（=20, 60, 80, 100, 200, 500）個漢字產生 m 個限制條件（公式 1），再由線性規劃方法計算出參數值，並利用這些參數來檢視所有 8593 筆測試漢字聲符判別之準確率。我們反覆十次上述的實驗，用以繪製盒鬚圖(Box-Whisker Plot)得到如圖二之結果。



圖二、以發音相似度最佳化作爲聲符預測準確圖

如圖二所示，我們可以藉由最佳化方法找到與前節所得準確率相當的相似度參數，甚至於在少數已知聲符的漢字訓練資料，如 40 或 60 筆時，即有相當好的結果，超越修正版 89.39%的準確率，所得參數值與人工定義的參數設定方式接近，如聲母參數 ub>a>b>c,d>0.1 等關係，韻母參數 1>x>y,z>lb 等關係（如表六所示）。事實上聲符預測準確率最高 89.98%，可由 60 筆已知聲符的形聲字所產生的限制條件求出，然當訓練資料（限制條件）增加時，準確率並無上升的情形，甚至在 500 時，反而有下降趨勢（最佳測試準確率降至 87%及 81.5%）。推測主要的原因在於聲符與形聲字發音相似度大於非聲符構件與形聲字發音相似度僅是一個通則，仍有相當多例外的情形。例如，冶、洛、債、時、枸、荼等字，前二者產生 $y<=lb+\varepsilon$ 的限制，中二者則產生 $b<=d+\varepsilon$ 的限制，後二者則產生 $z<=lb+\varepsilon$ 的限制條件。在少量的訓練資料時，這些限制條件的影響可能只是影響一二個變數大小關係，但是在較多的訓練資料時，將可能同時影響數個變數，最終是聲母參數均相同（ub=a=b=c=d），韻母參數均相同（x=y=z=lb）時，反而可以讓目標函數的值較小。因此我們看到當訓練資料 500 筆時，即使是最佳一組測試準確率，也僅得聲母參數均相同的結果。

表六、不同筆訓練資料所得最佳測試準確率及其聲母、韻母參數

訓練資料準確率	90.0%	98.0%	98.0%	96.0%	94.0%	98.0%	97.0%
測試資料準確率	89.0%	89.8%	90.0%	89.5%	90.0%	90.0%	88.0%
ub	0.97	0.82	0.94	0.93	1.00	0.66	0.93
a	0.73	0.55	0.63	0.73	0.84	0.89	0.10
b	§0.10	§0.10	0.60	0.76	§0.37	0.50	0.10
c	§0.46	§0.67	0.00	0.10	§0.42	0.10	0.10
d	§0.10	§0.40	0.55	0.19	§0.58	0.10	0.10
x	0.63	0.70	0.72	0.83	0.64	0.89	0.93
y	0.10	¶0.22	¶0.38	¶0.33	¶0.16	0.18	0.10
z	0.10	¶0.34	¶0.51	¶0.52	¶0.41	♮0.10	♮0.10
lb	0.10	0.16	0.10	0.13	0.10	♮0.13	♮0.12

Affected by constraints § b<=c or d, ¶ y<=z, ♮z<=lb

4.3 機率分佈比較法

第三部份的實驗目的在於了解機率分佈比較法對於在判別漢字的聲符之效能。如表七所示，以聲母分佈、韻母分佈以及聲調分佈個別強度做爲聲符預測，都有一定的準確度（分別爲 92.8%、97.5%及 95.9%），其中又以韻母的分佈是三者當中最爲有效方式。整體來說，三種分佈一起考量的結果有最佳的效果，針對 9593 筆形聲字，其中有 9413 筆正確，180 筆錯誤，準確率 98.1%。

表七、機率分佈比較法判別準確率。

	正確	錯誤	準確率
聲	8910	683	92.80
韻	9362	231	97.50
調	9207	386	95.90
聲+韻+調	9413	180	**98.10**

爲了解判定錯誤發生的可能原因，我們列出錯誤例子如表八。這些例子顯示，機率分佈比較法發生錯誤多在在於構件間的強度太過接近，尤其是兩個構件均爲部首的情形，然而若就發音相似度比較法而言應該可以正確判斷其聲符。因此也可以考慮結合機率分佈比較法與發音相似度比較法，對於形聲字進行聲符的判斷測試。

表八、機率分佈比較法錯誤例子

word	Phonetic notation	Elements	Correct Phonetic Symbol	Sum of KL divergence of each element
扣	ㄎㄡ4 (kou)	扌口	口	扌 0.0756 > 口 0.0692
沐	ㄇㄨ4 (mu)	氵木	木	氵 0.0284 > 木 0.0218
孟	ㄇㄥ4 (meng)	子皿	皿	子 0.6303 > 皿 0.2763
忝	ㄊㄧㄢ3 (tian)	天小	天	天 1.2455 < 小 1.8219
所	ㄙㄨㄛ3 (suo)	戶斤	戶	戶 0.7605 < 斤 0.9602
旺	ㄨㄤ4 (wang)	日王	王	日 0.1051 > 王 0.0904

5. 結論及未來研究

本篇論文主要目的是藉由對形聲字的分析研究，找出漢字與其聲符構件原字之間的關係。在第一階段我們針對漢字形聲字聲符的標記，除了採用中文系研究生的人力標記之外，同時也提出三種自動判別的方式，用以加速形聲字聲符的標記工作。實驗顯示，不論是人工制訂的相似度參數或是最佳化方式所得的參數，預測準確率大約可以做到九成。這是否是本篇論文所使用的最佳化方法的不足，又或是發音相似度比較法對於聲符預測的極限，仍尚待進一步的研究。另外一方面，以每個部件發音的分佈集中與否做爲聲符的判斷，則有高達九成八的準確率，則是相當有用的資訊。

雖然形聲字在聲符標注完成後，預測聲符的需求即消失不在，但是透過發音相似度最佳化方法所得的聲母，韻母相似度參數或許有助於未來漢字字音處理的研究，同時部件發音強度也可做爲漢字教學順序參考，仍有相當的重要性。未來我們將以持續以挑戰漢字的發音規則知識庫爲出發，除解析漢字發音規則外，也希望以此項研究爲出發，發展一套漢字學習的順序，讓使用者可用較少的學習時間，有效率認識更多漢字。

致謝

本論文的完成感謝陳怡如、葉博榮、鍾哲宇、趙婕妤等人的幫助。

參考資料

許慎撰，段玉裁注(1988)。《說文解字注》，台北藝文印書館。

莊德明、謝清俊(2005，1 月)。漢字構形資料庫的建置與應用，漢字與全球化國際學術研討會，台北。

莊德明、鄧賢瑛(2008，8 月)。文字學入口網站的規畫，第四屆中國文字學國際學術研討會，山東煙台。

附錄A:聲母相似度比較表

聲	ㄅ	ㄆ	ㄇ	ㄈ	ㄉ	ㄊ	ㄋ	ㄌ	ㄍ	ㄎ	ㄏ	ㄐ	ㄑ	ㄒ	ㄓ	ㄔ	ㄕ	ㄖ	ㄗ	ㄘ	ㄙ
ㄅ	1	0.9	0.8	0.1/0.8	0.5	0.1	0.1	0.1	0.5	0.5	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
ㄆ		1	0.8	0.1/0.8	0.1	0.5	0.1	0.1	0.1	0.5	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
ㄇ			1	0.1/0.8	0.1	0.1	0.5	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
ㄈ				1	0.1	0.1	0.1	0.1	0.1	0.1	0.5	0.1	0.1	0.5	0.1	0.1	0.5	0.1	0.1	0.1	0.5
ㄉ					1	0.9	0.8	0.8	0.5	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
ㄊ						1	0.8	0.8	0.1	0.5	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
ㄋ							1	0.8	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
ㄌ								1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
ㄍ									1	0.9	0.8	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
ㄎ										1	0.9	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
ㄏ											1	0.9	0.1	0.1	0.5	0.1	0.1	0.1	0.1	0.1	0.5
ㄐ												1	0.9	0.8	0.1/0.7	0.1/0.7	0.1/0.7	0.1/0.7	0.5/0.7	0.1/0.7	0.1/0.7
ㄑ													1	0.8	0.1/0.7	0.5/0.7	0.1/0.7	0.1/0.7	0.1/0.7	0.5/0.7	0.1/0.7
ㄒ														1	0.1/0.7	0.1/0.7	0.5/0.7	0.1/0.7	0.1/0.7	0.1/0.7	0.5/0.7
ㄓ															1	0.9	0.8	0.8	0.5/0.7	0.1/0.7	0.1/0.7
ㄔ																1	0.8	0.1/0.7	0.1/0.7	0.5/0.7	0.1/0.7
ㄕ																	1	0.8	0.1/0.7	0.1/0.7	0.5/0.7
ㄖ																		1	0.1/0.7	0.1/0.7	0.1/0.7
ㄗ																			1	0.9	0.8
ㄘ																				1	0.8
ㄙ																					1

附錄B:韻母相似度比較表

韻母		ㄓ 空韻	ㄧ i	ㄨ u	ㄩ y	ㄚ a	ㄛ o	ㄜ ɤ	ㄝ e	ㄞ ai	ㄟ ei	ㄠ au	ㄡ ou	ㄢ an	ㄣ ən	ㄤ aŋ	ㄥ əŋ	ㄦ ər	ㄧㄚ ia	ㄧㄛ io	ㄧㄝ ie	ㄧㄞ iai	ㄧㄠ iau	ㄧㄡ iou	ㄧㄢ ian	ㄧㄣ iən	ㄧㄤ iaŋ	ㄧㄥ iəŋ	ㄨㄚ ua	ㄨㄛ uo	ㄨㄞ uai	ㄨㄟ uei	ㄨㄢ uan	ㄨㄣ uən	ㄨㄤ uaŋ	ㄨㄥ uəŋ	ㄩㄝ ye	ㄩㄢ yan	ㄩㄣ yən	ㄩㄥ yəŋ
ㄓ	空韻	1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
ㄧ	i		1	0.1	0.1	0.1	0.1	0.1	0.1	0.5	0.5	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
ㄨ	u			1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.5	0.5	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.1	0.1	0.1	0.1
ㄩ	y				1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.5	0.5	0.5	0.5
ㄚ	a					1	0.1	0.1	0.1	0.5	0.1	0.5	0.1	0.5	0.1	0.5	0.1	0.1	0.8	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.8	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
ㄛ	o						1	0.1	0.1	0.1	0.1	0.1	0.5	0.1	0.1	0.1	0.1	0.1	0.1	0.8	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.8	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
ㄜ	ɤ							1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
ㄝ	e								1	0.1	0.5	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.8	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.8	0.1	0.1	0.1
ㄞ	ai									1	0.5	0.5	0.1	0.5	0.1	0.5	0.1	0.1	0.1	0.1	0.1	0.8	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.8	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
ㄟ	ei										1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.8	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
ㄠ	au											1	0.5	0.5	0.1	0.5	0.1	0.1	0.1	0.1	0.1	0.8	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.8	0.1	0.1	0.1	0.1	0.1	0.1	0.1
ㄡ	ou												1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.8	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
ㄢ	an													1	0.5	0.5	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.8	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.8	0.1	0.1	0.1	0.1	0.8	0.1	0.1
ㄣ	ən														1	0.1	0.5	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.8	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.8	0.1	0.1	0.1	0.1	0.8	0.1
ㄤ	aŋ															1	0.5	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.8	0.1	0.1	0.1	0.1	0.1	0.1	0.8	0.1	0.1	0.1	0.1	0.1	0.1
ㄥ	əŋ																1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.8	0.1	0.1	0.1	0.1	0.1	0.1	0.8	0.1	0.1	0.1	0.8
ㄦ	ər																	1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
ㄧㄚ	ia																		1	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.8	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
ㄧㄛ	io																			1	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.1	0.8	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
ㄧㄝ	ie																				1	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
ㄧㄞ	iai																					1	0.5	0.5	0.5	0.5	0.5	0.5	0.1	0.8	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
ㄧㄠ	iau																						1	0.5	0.5	0.5	0.5	0.5	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
ㄧㄡ	iou																							1	0.5	0.5	0.5	0.5	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
ㄧㄢ	ian																								1	0.5	0.5	0.5	0.1	0.1	0.1	0.1	0.8	0.1	0.1	0.1	0.1	0.8	0.1	0.1
ㄧㄣ	iən																									1	0.5	0.5	0.1	0.1	0.1	0.1	0.1	0.8	0.1	0.1	0.1	0.1	0.8	0.1
ㄧㄤ	iaŋ																										1	0.5	0.1	0.1	0.1	0.1	0.1	0.1	0.8	0.1	0.1	0.1	0.1
ㄧㄥ	iəŋ																											1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.8	0.1	0.1	0.1	0.8
ㄨㄚ	ua																												1	0.5	0.5	0.5	0.5	0.5	0.5	0.1	0.1	0.1	0.1
ㄨㄛ	uo																													1	0.5	0.5	0.5	0.5	0.5	0.1	0.1	0.1	0.1
ㄨㄞ	uai																														1	0.5	0.5	0.5	0.5	0.1	0.1	0.1	0.1
ㄨㄟ	uei																															1	0.5	0.5	0.5	0.1	0.1	0.1	0.1
ㄨㄢ	uan																																1	0.5	0.5	0.1	0.8	0.1	0.1
ㄨㄣ	uən																																	1	0.5	0.5	0.1	0.1	0.8	0.1
ㄨㄤ	uaŋ																																		1	0.5	0.1	0.1	0.1	0.8
ㄨㄥ	uəŋ																																			1	0.5	0.5	0.5	0.5
ㄩㄝ	ye																																				1	0.5	0.5	0.5
ㄩㄢ	yan																																					1	0.5	0.5
ㄩㄣ	yən																																						1	0.5
ㄩㄥ	yəŋ																																							1

The Association for Computational Linguistics and Chinese Language Processing

(new members are welcomed)

Aims：

1. To conduct research in computational linguistics.
2. To promote the utilization and development of computational linguistics.
3. To encourage research in and development of the field of Chinese computational linguistics both domestically and internationally.
4. To maintain contact with international groups who have similar goals and to cultivate academic exchange.

Activities：

1. Holding the Republic of China Computational Linguistics Conference (ROCLING) annually.
2. Facilitating and promoting academic research, seminars, training, discussions, comparative evaluations and other activities related to computational linguistics.
3. Collecting information and materials on recent developments in the field of computational linguistics, domestically and internationally.
4. Publishing pertinent journals, proceedings and newsletters.
5. Setting of the Chinese-language technical terminology and symbols related to computational linguistics.
6. Maintaining contact with international computational linguistics academic organizations.
7. Dealing with various other matters related to the development of computational linguistics.

To Register：

Please send application to:

The Association for Computational Linguistics and Chinese Language Processing
Institute of Information Science, Academia Sinica
128, Sec. 2, Academy Rd., Nankang, Taipei 11529, Taiwan, R.O.C.

payment： Credit cards(please fill in the order form), cheque, or money orders.

Annual Fees：

regular/overseas member： NT$ 1,000 (US$50.-)
group membership： NT$20,000 (US$1,000.-)
life member：ten times the annual fee for regular/ group/ overseas members

Contact：

Address： The Association for Computational Linguistics and Chinese Language Processing
Institute of Information Science, Academia Sinica
128, Sec. 2, Academy Rd., Nankang, Taipei 11529, Taiwan, R.O.C.

Tel.：886-2-2788-3799 ext. 1502 Fax：886-2-2788-1638

E-mail: aclclp@hp.iis.sinica.edu.tw Web Site: http://www.aclclp.org.tw

Please address all correspondence to Miss Qi Huang, or Miss Abby Ho

The Association for Computational Linguistics and Chinese Language Processing

Membership Application Form

Member ID#： _____

Name： _____ Date of Birth： _____

Country of Residence： _____ Province/State： _____

Passport No.： _____ Sex: _____

Education(highest degree obtained)： _____

Work Experience： _____

_____

Present Occupation： _____

Address： _____

_____

Email Add： _____

Tel. No： _____ Fax No： _____

Membership Category： ☐ Regular Member ☐ Life Member

Date： ____/____/____（Y-M-D）

Applicant's Signature：

Remarks： Please indicated clearly in which membership category you wish to register, according to the following scale of annual membership dues：

Regular Member ： US$ 50.- （NT$ 1,000）

Life Member ： US$500.-（NT$10,000）

Please feel free to make copies of this application for others to use.

Committee Assessment：

中華民國計算語言學學會

宗旨：

（一） 從事計算語言學之研究
（二） 推行計算語言學之應用與發展
（三） 促進國內外中文計算語言學之研究與發展
（四） 聯繫國際有關組織並推動學術交流

活動項目：

（一）定期舉辦中華民國計算語言學學術會議（Rocling）
（二）舉行有關計算語言學之學術研究講習、訓練、討論、觀摩等活動項目
（三）收集國內外有關計算語言學知識之圖書及最新發展之資料
（四）發行有關之學術刊物，論文集及通訊
（五）研定有關計算語言學專用名稱術語及符號
（六）與國際計算語言學學術機構聯繫交流
（七）其他有關計算語言發展事項

報名方式：

1. 入會申請書：請至本會網頁下載入會申請表，填妥後郵寄或E-mail至本會
2. 繳交會費：劃撥：帳號：19166251，戶名：中華民國計算語言學學會
   信用卡：請至本會網頁下載信用卡付款單

年費：

終身會員：	10,000.-	（US$ 500.-）
個人會員：	1,000.-	（US$ 50.-）
學生會員：	500.-	（限國內學生）
團體會員：	20,000.-	（US$ 1,000.-）

連絡處：

地址：台北市115南港區研究院路二段128號 中研院資訊所(轉)
電話：(02) 2788-3799 ext.1502 傳真：(02) 2788-1638
E-mail：aclclp@hp.iis.sinica.edu.tw 網址: http://www.aclclp.org.tw
連絡人：黃琪 小姐、何婉如 小姐

中華民國計算語言學學會
個人會員入會申請書

<table>
<tr><td>會員類別</td><td colspan="2">□終身 □個人 □學生</td><td>會員編號</td><td colspan="2">（由本會填寫）</td></tr>
<tr><td rowspan="2">姓　　名</td><td rowspan="2"></td><td rowspan="2">性別</td><td rowspan="2"></td><td>出生日期</td><td>年　月　日</td></tr>
<tr><td>身分證號碼</td><td></td></tr>
<tr><td>現　　職</td><td></td><td>學　　歷</td><td colspan="2"></td></tr>
<tr><td>通訊地址</td><td colspan="5">□□□</td></tr>
<tr><td>戶籍地址</td><td colspan="5">□□□</td></tr>
<tr><td>電　　話</td><td colspan="2"></td><td>E-Mail</td><td colspan="2"></td></tr>
<tr><td colspan="6">申請人:　　（簽章）

中　華　民　國　　年　　月　　日</td></tr>
</table>

審查結果:

1. 年費：

   終身會員：　10,000.-
   個人會員：　1,000.-
   學生會員：　500.-（限國內學生）
   團體會員：　20,000.-

2. 連絡處：

   地址：台北市南港區研究院路二段128號 中研院資訊所(轉)
   電話：(02) 2788-3799　ext.1502 傳真：(02) 2788-1638
   E-mail：aclclp@hp.iis.sinica.edu.tw　　網址: http://www.aclclp.org.tw
   連絡人：黃琪 小姐、何婉如 小姐

3. 本表可自行影印

The Association for Computational Linguistics and Chinese Language Processing (ACLCLP)

PAYMENT FORM

Name :_____(Please print) Date: _____

Please debit my credit card as follows: US$_____

❑ VISA CARD ❑ MASTER CARD ❑ JCB CARD Issue Bank:_____

Card No.: _____- _____ - _____ - _____ Exp. Date:_____

3-digit code: _____ (on the back card, inside the signature area, the last three digits)

CARD HOLDER SIGNATURE : _____

Tel.: _____ E-mail: _____

Add: _____

PAYMENT FOR

US$ _____ ❑ Computational Linguistics & Chinese Languages Processing (CLCLP)

Quantity Wanted: _____

US$ _____ ❑ Publications:_____

US$ _____ ❑ Text Corpora: _____

US$ _____ ❑ Speech Corpora:_____

US$ _____ ❑ Others: _____

US$ _____ ❑Life Member Fee ❑ New Member ❑Renew

US$ _____ = Total

Fax : 886-2-2788-1638 or Mail this form to :

ACLCLP

% Institute of Information Science, Academia Sinica

R502, 128, Sec.2, Academia Rd., Nankang, Taipei 115, Taiwan

E-mail: aclclp@hp.iis.sinica.edu.tw

Website: http://www.aclclp.org.tw

中 華 民 國 計 算 語 言 學 學 會
信用卡付款單

姓名: _____(請以正楷書寫)　日期:： _____

卡別：❑ VISA CARD ❑ MASTER CARD ❑ JCB CARD　發卡銀行：_____

卡號:_____-_____-_____- _____　有效日期：_____

卡片後三碼：_____（卡片背面簽名欄上數字後三碼）

持卡人簽名：　_____(簽名方式請與信用卡背面相同)

通訊地址：_____

聯絡電話：_____　E-mail：_____

備註：為順利取得信用卡授權，請提供與發卡銀行相同之聯絡資料。

付款內容及金額：

NT$_____❑ 中文計算語言學期刊(IJCLCLP)

NT$_____❑ 中研院詞庫小組技術報告

NT$_____❑ 中文（新聞）語料庫

NT$_____❑ 平衡語料庫

NT$_____❑ 中文詞庫八萬目

NT$_____❑ 中文句結構樹資料庫

NT$_____❑ 平衡語料庫詞集及詞頻統計

NT$_____❑ 中英雙語詞網

NT$_____❑ 中英雙語知識庫

NT$_____❑ 語音資料庫_____

NT$_____❑ 會員年費　❑續會　❑新會員　❑終身會員

NT$_____❑ 其他: _____

NT$_____= 合計

填妥後請傳真至 02-27881638 或郵寄至:
115台北市南港區研究院路2段128號中研院資訊所(轉)中華民國計算語言學學會 收
E-mail: aclclp@hp.iis.sinica.edu.tw
Website: http://www.aclclp.org.tw

Publications of the Association for Computational Linguistics and Chinese Language Processing

		Surface	AIR (US&EURP)	AIR (ASIA)	VOLUME	AMOUNT
1.	no.92-01, no. 92-04(合訂本) ICG 中的論旨角色與 A Conceptual Structure for Parsing Mandarin -- Its Frame and General Applications--	US$ 9	US$ 19	US$15	_____	_____
2.	no.92-02 V-N 複合名詞討論篇 & 92-03 V-R 複合動詞討論篇	12	21	17	_____	_____
3.	no.93-01 新聞語料庫字頻統計表	8	13	11	_____	_____
4.	no.93-02 新聞語料庫詞頻統計表	18	30	24	_____	_____
5.	no.93-03 新聞常用動詞詞頻與分類	10	15	13	_____	_____
6.	no.93-05 中文詞類分析	10	15	13	_____	_____
7.	no.93-06 現代漢語中的法相詞	5	10	8	_____	_____
8.	no.94-01 中文書面語頻率詞典（新聞語料詞頻統計）	18	30	24	_____	_____
9.	no.94-02 古漢語字頻表	11	16	14	_____	_____
10.	no.95-01 注音檢索現代漢語字頻表	8	13	10	_____	_____
11.	no.95-02/98-04 中央研究院平衡語料庫的內容與說明	3	8	6	_____	_____
12.	no.95-03 訊息爲本的格位語法與其剖析方法	3	8	6	_____	_____
13.	no.96-01「搜」文解字－中文詞界研究與資訊用分詞標準	8	13	11	_____	_____
14.	no.97-01 古漢語詞頻表（甲）	19	31	25	_____	_____
15.	no.97-02 論語詞頻表	9	14	12	_____	_____
16.	no.98-01 詞頻詞典	18	30	26	_____	_____
17.	no.98-02 Accumulated Word Frequency in CKIP Corpus	15	25	21	_____	_____
18.	no.98-03 自然語言處理及計算語言學相關術語中英對譯表	4	9	7	_____	_____
19.	no.02-01 現代漢語口語對話語料庫標註系統說明	8	13	11	_____	_____
20.	Computational Linguistics & Chinese Languages Processing (One year) (Back issues of *IJCLCLP*: US$ 20 per copy)	---	100	100	_____	_____
21.	Readings in Chinese Language Processing	25	25	21	_____	_____
				TOTAL	_____	_____

10% member discount: _____Total Due:_____

- **OVERSEAS USE ONLY**
- PAYMENT： ☐ Credit Card (Preferred)
  ☐ Money Order or Check payable to "The Association for Computation Linguistics and Chinese Language Processing " or “中華民國計算語言學學會”
- E-mail：aclclp@hp.iis.sinica.edu.tw

Name (please print): _____ Signature: _____

Fax: _____ E-mail: _____

Address：_____

中華民國計算語言學學會

相關出版品價格表及訂購單

編號	書目	會　員	非會員	冊數	金額
1.	no.92-01, no. 92-04 (合訂本)　ICG 中的論旨角色 與 A conceptual Structure for Parsing Mandarin--its Frame and General Applications--	NT$ 80	NT$ 100	_____	_____
2.	no.92-02, no. 92-03 (合訂本) V-N 複合名詞討論篇 與V-R 複合動詞討論篇	120	150	_____	_____
3.	no.93-01　新聞語料庫字頻統計表	120	130	_____	_____
4.	no.93-02　新聞語料庫詞頻統計表	360	400	_____	_____
5.	no.93-03　新聞常用動詞詞頻與分類	180	200	_____	_____
6.	no.93-05　中文詞類分析	185	205	_____	_____
7.	no.93-06　現代漢語中的法相詞	40	50	_____	_____
8.	no.94-01　中文書面語頻率詞典（新聞語料詞頻統計）	380	450	_____	_____
9.	no.94-02　古漢語字頻表	180	200	_____	_____
10.	no.95-01　注音檢索現代漢語字頻表	75	85	_____	_____
11.	no.95-02/98-04　中央研究院平衡語料庫的內容與說明	75	85	_____	_____
12.	no.95-03　訊息爲本的格位語法與其剖析方法	75	80	_____	_____
13.	no.96-01「搜」文解字－中文詞界研究與資訊用分詞標準	110	120	_____	_____
14.	no.97-01　古漢語詞頻表（甲）	400	450	_____	_____
15.	no.97-02　論語詞頻表	90	100	_____	_____
16	no.98-01　詞頻詞典	395	440	_____	_____
17.	no.98-02　Accumulated Word Frequency in CKIP Corpus	340	380	_____	_____
18.	no.98-03 自然語言處理及計算語言學相關術語中英對譯表	90	100	_____	_____
19.	no.02-01 現代漢語口語對話語料庫標註系統說明	75	85	_____	_____
20	論文集 COLING 2002 紙本	100	200	_____	_____
21.	論文集 COLING 2002 光碟片	300	400	_____	_____
22.	論文集 COLING 2002 Workshop 光碟片	300	400	_____	_____
23.	論文集 ISCSLP 2002 光碟片	300	400	_____	_____
24.	交談系統暨語境分析研討會講義（中華民國計算語言學學會1997第四季學術活動）	130	150	_____	_____
25.	中文計算語言學期刊（一年四期）　年份：_____（過期期刊每本售價500元）	---	2,500	_____	_____
26.	Readings of Chinese Language Processing	675	675	_____	_____
27.	剖析策略與機器翻譯 1990	150	165	_____	_____
			合　計	_____	_____

※ 此價格表僅限國內 (台灣地區) 使用

劃撥帳戶：中華民國計算語言學學會　　劃撥帳號：19166251

聯絡電話：(02) 2788-3799 轉1502

聯絡人： 黃琪 小姐、何婉如 小姐　　E-mail:aclclp@hp.iis.sinica.edu.tw

訂購者：_____ 收據抬頭：_____

地　　址：_____

電　　話：_____ E-mail: _____

Information for Authors

International Journal of Computational Linguistics and Chinese Language Processing (IJCLCLP) invites submission of original research papers in the area of computational linguistics and speech/text processing of natural language. All papers must be written in English or Chinese. Manuscripts submitted must be previously unpublished and cannot be under consideration elsewhere. Submissions should report significant new research results in computational linguistics, speech and language processing or new system implementation involving significant theoretical and/or technological innovation. The submitted papers are divided into the categories of regular papers, short paper, and survey papers. Regular papers are expected to explore a research topic in full details. Short papers can focus on a smaller research issue. And survey papers should cover emerging research trends and have a tutorial or review nature of sufficiently large interest to the Journal audience. There is no strict length limitation on the regular and survey papers. But it is suggested that the manuscript should not exceed 40 double-spaced A4 pages. In contrast, short papers are restricted to no more than 20 double-spaced A4 pages. All contributions will be anonymously reviewed by at least two reviewers.

Copyright : It is the author's responsibility to obtain written permission from both author and publisher to reproduce material which has appeared in another publication. Copies of this permission must also be enclosed with the manuscript. It is the policy of the CLCLP society to own the copyright to all its publications in order to facilitate the appropriate reuse and sharing of their academic content. A signed copy of the IJCLCLP copyright form, which transfers copyright from the authors (or their employers, if they hold the copyright) to the CLCLP society, will be required before the manuscript can be accepted for publication. The papers published by IJCLCLP will be also accessed online via the IJCLCLP official website and the contracted electronic database services.

Style for Manuscripts: The paper should conform to the following instructions.

1. Typescript: Manuscript should be typed double-spaced on standard A4 (or letter-size) white paper using size of 11 points or larger.

2. Title and Author: The first page of the manuscript should consist of the title, the authors' names and institutional affiliations, the abstract, and the corresponding author's address, telephone and fax numbers, and e-mail address. The title of the paper should use normal capitalization. Capitalize only the first words and such other words as the orthography of the language requires beginning with a capital letter. The author's name should appear below the title.

3. Abstracts and keywords: An informative abstract of not more than 250 words, together with 4 to 6 keywords is required. The abstract should not only indicate the scope of the paper but should also summarize the author's conclusions.

4. Headings: Headings for sections should be numbered in Arabic numerals (i.e. 1.,2....) and start form the left-hand margin. Headings for subsections should also be numbered in Arabic numerals (i.e. 1.1. 1.2...).

5. Footnotes: The footnote reference number should be kept to a minimum and indicated in the text with superscript numbers. Footnotes may appear at the end of manuscript

6. Equations and Mathematical Formulas: All equations and mathematical formulas should be typewritten or written clearly in ink. Equations should be numbered serially on the right-hand side by Arabic numerals in parentheses.

7. References: All the citations and references should follow the APA format. The basic form for a reference looks like

Authora, A. A., Authorb, B. B., & Authorc, C. C. (Year). Title of article. *Title of Periodical, volume number*(issue number), pages.

Here shows an example.

Scruton, R. (1996). The eclipse of listening. *The New Criterion, 15*(30), 5-13.

The basic form for a citation looks like (Authora, Authorb, and Authorc, Year). Here shows an example. (Scruton, 1996).

Please visit the following websites for details.

(1) APA Formatting and Style Guide (http://owl.english.purdue.edu/owl/resource/560/01/)

(2) APA Stytle (http://www.apastyle.org/)

No page charges are levied on authors or their institutions.

Final Manuscripts Submission: If a manuscript is accepted for publication, the author will be asked to supply final manuscript in MS Word or PDF files to clp@hp.iis.sinica.edu.tw

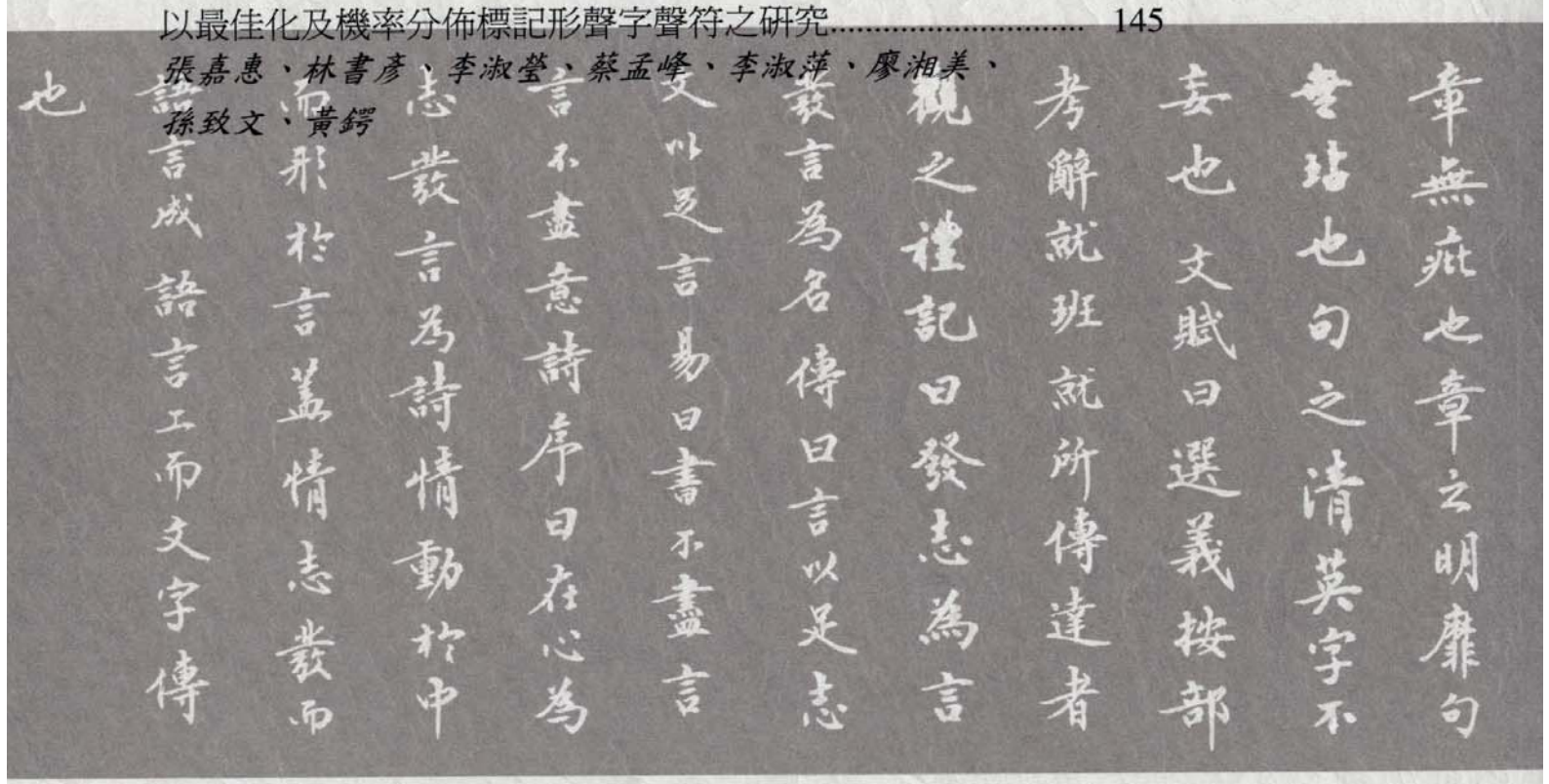
Online Submission: http://www.aclclp.org.tw/journal/submit.php

Please visit the IJCLCLP Web page at http://www.aclclp.org.tw/journal/index.php

International Journal of
Computational Linguistics & Chinese Language Processing
vol.15 no.2 June 2010

Contents

Papers



ISSN: 1027-376X

The Association for Computational Linguistics and Chinese Language Processing