An Omni-font Gurmukhi to Shahmukhi Transliteration System

Gurpreet Singh LEHAL[1] *Tejinder Singh SAINI*[2] *Savleen Kaur CHOWDHARY*[3]
(1) DCS, Punjabi University, Patiala
(2) ACTDPL, Punjabi University, Patiala
(3) CEC, Chandigarh Group of Colleges, Landran, Mohali
gslehal@gmail.com, tej@pbi.ac.in, iridiumsavleen@gmail.com

ABSTRACT

This paper describes a font independent Gurmukhi-to-Shahmukhi transliteration system. Even though Unicode is gaining popularity, but still there is lot of material in Punjabi, which is available in ASCII based fonts. A problem with ASCII fonts for Punjabi is there is no standardisation of mapping of Punjabi characters and a Gurmukhi character may be internally mapped to different keys in different Punjabi fonts. In fact there are more than 40 mapping tables in use for the commonly used Punjabi fonts. Thus there is an urgent need to convert the ASCII fonts to standard mapping, such as Unicode without any loss of information. Already many such font converters have been developed and are available online. But one limitation is that, all these systems need manual intervention in which the user has to know the name of source font. In the first stage, we have proposed a statistical model for automatic font detection and conversion into Unicode. Our system supports around 225 popular Gurmukhi font encodings. The ASCII to Unicode conversion accuracy of the system is 99.73% at word level with TOP1 font detection. The second stage is conversion of Gurmukhi to Shahmukhi at high accuracy. The proposed Gurmukhi to Shahmukhi transliteration system can transliterate any Gurmukhi text to Shahmukhi at more than 98.6% accuracy at word level.

KEYWORDS: n-gram language model, Shahmukhi, Gurmukhi, Machine Transliteration, Punjabi, Font, Font detection, font Conversion, Unicode

1 Introduction

There are thousands of fonts used for publishing text in Indian languages. Punjabi (Gurmukhi) alone has more than 225 popular fonts which are still in use along with Unicode. Though online web content has shown the sign of migration from legacy-font to Unicode but books, magazine, news paper and other publishing industry is still working with ASCII based fonts. They have not adopted Unicode due to following reasons:

- Lack of awareness of Unicode standard
- People resist to change, due to Unicode typing issues and little support of Unicode in publishing software they are working with
- Less availability of Unicode fonts has shown very less verity of text representation

A problem with ASCII fonts for Punjabi is there is no standardisation of mapping of Punjabi characters and a Gurmukhi character may be internally mapped to different keys in different Punjabi fonts. For example, the word ਪੰਜਾਬੀ is internally stored at different keys as shown in figure 1. Therefore, there is an urgent need to develop a converter to convert text in ASCII based fonts to Unicode without any loss of information. Already many such font converters have been developed and are available online. But one limitation with all these systems is that the user has to know the name of source font. This may not be a big issue but there are many occasions when a user may not be aware of the name of the ASCII font and in that case he cannot convert his text to Unicode. To overcome this hurdle, we have developed a font identification system, which automatically detects the font and then converts the text to Unicode.

0070+004D+006A+0077+0062+0049	in *Akhar*
0066+002E+0075+006A+0057+0067	in *Gold*
0070+00B5+006A+003B+0062+0049	in *AnandpurSahib*
0067+007A+0069+006B+0070+0068	in *Asees*
0050+005E+004A+0041+0042+0049	in *Sukhmani*
00EA+00B3+00DC+00C5+00EC+00C6	in *Satluj*

FIGURE 1– Internal representation of ਪੰਜਾਬੀ in different fonts

2 Omni-Font Detection and Conversion to Unicode

Our system supports around 225 popular Gurmukhi font encodings. Some of the popular fonts supported are *Akhar, Anmol Lipi, Chatrik, Joy, Punjabi, Satluj* etc. In fact, these fonts correspond to 41 keyboard mappings. It means if $k_1, k_2 .... k_{41}$ be the 41 keyboard mappings and $f_1, f_2 ...... f_{225}$ be the Gurmukhi fonts, then each of fonts f_i will belong to one of the keyboard mapping k_i. We could also have multiple fonts belonging to same keyboard map and fonts belonging to same keyboard map have same internal mappings for all the Gurmukhi

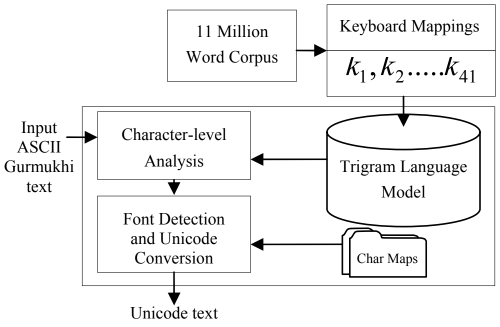

FIGURE 2– Model Components

characters. For example, *Akhar2010R* and *Joy* font belong to same keyboard map family. The problem is now reduced from 225 distinct fonts to just 41 group classes corresponding to each keyboard map. Therefore, our font detection problem is to classify the input text to one of these 41 keyboard mappings. It could be thought of as a 41 class pattern recognition problem. The proposed system is based on character-level trigram language model (see figure 2). We have trained the trigrams corresponding to each keyboard map. For training purpose a raw corpus of 11 million words has been used. To identify the font of a text, all the character trigrams are extracted and their probability is determined in each of the keyboard map. The keyboard map having maximum product of trigram probability is identified as the keyboard map corresponding to the input text.

2.1 Font Detection

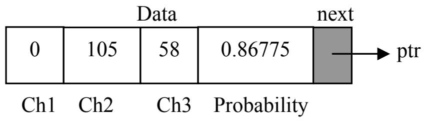

FIGURE 3– Tri-gram Character Node

The omni font detection problem is formulated as character level trigram language model. The single font has 255 character code points. Therefore, in a trigram model, we need to process 255x255x255=255^3 code points for a single keyboard map. But we are dealing with 41 distinct keyboard maps so the memory requirement will be further increased to process and hold 41x 255^3 code points. After detailed analysis we found that the array representation of this task is sparse in nature i.e. the majority of code points in omni fonts have zero values. We observed that each keyboard map has around 26,000 non-zero code points which is 0.156% of the original code points. Hence, to avoid sparse array formation, we have created trigram link list representation (see figure 3) having on an average 26,000 trigram nodes for only valid code points with respect to each keyboard map. The structure of the node is expressed in figure 2. The trigram probability of a word w^l having length l is:

$$w^l = \prod_{i=1}^{l} P(c_i \mid c_{i-1}, c_{i-2}) \qquad (1)$$

The total probability of input text sentence is:

$$w_{1,n} = \sum_{1}^{n} w^l \text{ and the keyboard map is detected by K= arg max } \sum_{1}^{k}\sum_{1}^{n} w^l \qquad (2)$$

2.2 Font Conversion

After the identification of the keyboard map, the handcrafted mapping table of identified font is used by the system to transform input text into Unicode characters and finally produce the output Unicode text. Special care has been given for transforming long vowels ਉ[ʊ], ਊ[u], ਓ[o], ਅ[ə], ਆ[ɑ], ਇ[ɪ], ਈ[i], ਏ[e], ਐ[æ], ਔ[Ɔ] and Gurmukhi short vowel sign ਿ◌[ɪ]. This is because in Unicode these nine independent vowels ਉ[ʊ], ਊ[u], ਓ[o], ਅ[ə], ਆ[ɑ], ਇ[ɪ], ਈ[i], ਏ[e], ਐ[æ], ਔ[Ɔ] with three bearer characters Ura ੳ[ʊ], Aira ਅ [ə] and Iri ੲ[ɪ] have single code points and need to be mapped accordingly as shown in figure 4. There are no explicit code points for half characters in Unicode. They will be generated automatically by the Unicode rendering system when it finds special symbol called halant or Viram [੍]. Therefore, Gurmukhi transformation of subjoined consonants is performed by prefixing halant ੍ symbol along with the respective consonant.

Long Vowels	ਅ[ə] + ਾ → ਆ[ɑ]	ੳ + ੁ → ਉ[ʊ]	ੲ + ਿ → ਇ[ɪ]
	ਅ[ə] + ੈ → ਐ[æ]	ੳ + ੂ → ਊ[u]	ੲ + ੀ → ਈ[i]
	ਅ[ə] + ੌ → ਔ[ɔ]	ੳ + ੋ → ਓ[o]	ੲ + ੇ → ਏ[e]
Short Vowel [ਿ]	ਕ[k] + ਿ → ਕਿ		
Nukta Symbol	ਸ + ਼ = ਸ਼; ਖ + ਼ = ਖ਼	ਗ + ਼ = ਗ਼; ਜ + ਼ = ਜ਼;	ਫ + ਼ = ਫ਼; ਲ + ਼ = ਲ਼;
Subjoined Consonants	ਨ+੍+ਹ = ਨ੍ਹ [n^h]	ਪ+੍+ਰ = ਪ੍ਰ [pr]	ਸ+੍+ਵ = ਸ੍ਵ [sv]

FIGURE 4– Unicode Transforming Rules

The Unicode transformation becomes complex when Gurmukhi short vowel *Sihari* [ਿ] and subjoined consonants comes together at a single word position. For example, consider the word ਪ੍ਰਿੰਸ. In omni font *Sihari* [ਿ] come as first character. But according to Unicode it must be after the bearing consonant. Therefore, it must go after the subjoined consonant ੍+ਰ as shown in figure 5.

ਪ੍ਰਿੰਸ (Omni-Font) → ਿ ਪ ੍ ੰ ਸ

→ ਪ + ੍ + ਰ + ਿ + ੰ + ਸ → ਪ੍ਰਿੰਸ (Unicode)

FIGURE 5– Complex Unicode Transformation

3 Gurmukhi-to-Shahmukhi Transliteration System

The overall transliteration process is divided into three tasks as shown in figure 6.

3.1 Pre-Processing

The integration of omni font detection and conversion module enhances the scope, usefulness and utilization of this transliteration system. The Gurmukhi word is cleaned and prepared for transliteration by passing it through the Gurmukhi spell-checker and normalizing it according to the Shahmukhi spellings and pronunciation as shown in Table 1.

Sr.	Gurmukhi Word	Spell-Checker	Normalized	Shahmukhi
1	ਖੁਦਗਰਜ /khudgaraj/	ਖ਼ੁਦਗ਼ਰਜ਼ /k̲hudġaraz/	ਖ਼ੂਦਗ਼ਰਜ਼ /k̲hūdġaraz/	خودغرض
2	ਖੁਸ਼ੀ /khushī/	ਖ਼ੁਸ਼ੀ /k̲hushī/	ਖ਼ੂਸ਼ੀ /k̲hūshī/	خوشی
3	ਫਰੰਗੀ /pharṅgī/	ਫ਼ਰੰਗੀ /farṅgī/	ਫ਼ਰੰਗੀ /farṅgī/	فرنگی

TABLE 1– Spell-Checking and Normalization of Gurmukhi Words

3.2 Transliteration Engine

The normalised form of Gurmukhi word is first transliterated to Shahmukhi by using dictionary lookup. Dictionary lookup is a limited but effective and fast method for handling the complex spelling words of Shahmukhi and transliteration of proper nouns. Therefore, a one-to-one

Gurmukhi-Shahmukhi dictionary resource has been created and used for directly transliterating frequently occurring Gurmukhi words and transliterating Gurmukhi words with typical Shahmukhi spellings.

Gurmukhi Stemmer: In our case, stemming is primarily a process of suffix removal. A list of common suffixes has been created. We have taken only the most common Gurmukhi suffixes such as ੋਂ, ਓਂ, ਿਓਂ, ੀਂ, ੇ etc. The Shahmukhi transliteration of these suffixes is stored in the suffix list. Thus if the word ਸਕੂਲੋਂ is not found, we search the suffix list and find suffix ੋਂ. This suffix ੋਂ is removed from ਸਕੂਲੋਂ and the resultant word ਸਕੂਲ is then searched again in the dictionary. If found then the transliterated word اسکول is then appended with وں, which is the Shahmukhi transliteration of the suffix ੋਂ. Thus, the correct transliteration i.e. اسکولوں is achieved.

3.2.1 Rule-base Transliteration

The rule based transliteration is used when dictionary lookup and Gurmukhi stemmer failed to transliterate the input word. Using the direct grapheme based approach; Gurmukhi consonants and vowel are directly mapped to similar sounding Shahmukhi characters. In case of multiple equivalent Shahmukhi characters or one-to-many mappings, the most frequently occurring Shahmukhi character is selected. Thus, ਹ is mapped to ہ and ਜ਼ is mapped to ز. Besides, these simple mapping rules, some special pronunciation based rules have also been developed. For example, if two vowels in Gurmukhi come together, then Shahmukhi symbol Hamza is placed in between them. After simple character based mapping we resolve character ambiguity. That is, the Gurmukhi characters with multiple Shahmukhi mappings, all word forms using all possible mappings are generated and the word with the highest frequency of occurrence in the Shahmukhi word frequency list is selected. For example, consider the Gurmukhi word ਸਾਹਿਬ. It has two ambiguous character ਸ[s] and ਹ[h]. The system will generate all the possible forms and then choose the most frequent صاحب (6432) unigram as output as shown in the figure 7. Finally, the output word is inspected for correct spelling using Shahmukhi resources.

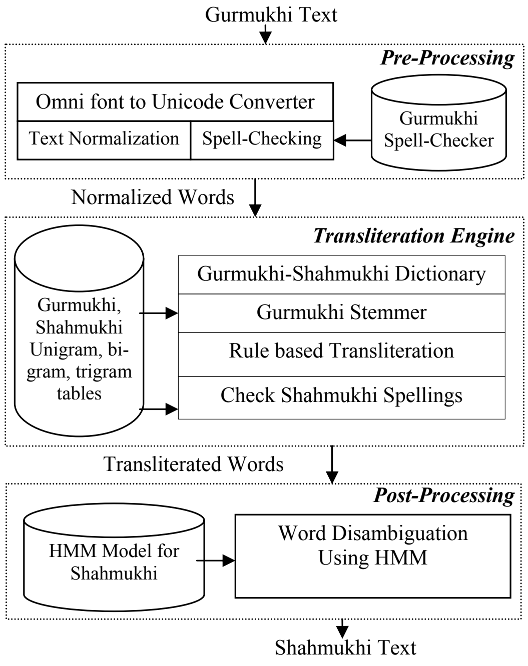

FIGURE 6– System Architecture

Char1	ਸ	→ س	ਸ → ص ਸ → ث	*All forms (with unigram frequency)*	
Char2	ਾ	→ ا		ساہب(0)	ساحب(0)
Char3	ਹ	→ ہ	ਹ → ح	صاہب(0)	صاحب(6432)
Char4	ਿ	→ nil		ثاہب(0)	ثاحب(0)
Char5	ਬ	→ب		*Selected →* *(most frequent)*	صاحب

FIGURE 7– Steps for handling Multiple Character Mapping

3.3 Post-Processing

The transliteration of Gurmukhi word ਹਾਲ has two Shahmukhi spellings with different senses as حال (state, condition, circumstance) and ہال (Hall; big room). In post-processing we have modelled 2nd order HMM for Shahmukhi word disambiguation where the number of HMM states are corresponding to each word in the sentence and their possible interpretations or ambiguity. The transition probabilities and inter-state observation probabilities are calculated as proposed by Thede and Harper (1999) for POS system.

4 Evaluation and Results

There are two main stages of our system first is font detection and Conversion of detected font to Unicode and the second stage is conversion of Gurmukhi text to Shahmukhi. The evaluation results of both the stages are:

Font Detection and Conversion Accuracy: In order to test font detection module we have a set of 35 popular Gurmukhi fonts. A randomly selected test data of having 194 words and having 962 characters (with spaces) is converted into different Gurmukhi fonts and used for system testing. The font detection module has correctly predicted 32 fonts. Hence, they have 100% conversion rate into Unicode. The remaining 3 fonts were not correctly recognized and confused with other fonts with which the character mapping table is almost similar. The fonts varied by only 2-3 code maps and hence the confusion. But this confusion did not create much problem in the subsequent Unicode and then Shahmukhi conversion.

Transliteration Accuracy: We have tested our system on more that 100 pages of text compiled from newspapers, books and poetry. The overall transliteration accuracy of this system is 98.6% at word level, which is quite high. In the error analysis we found that this system fails to produce good transliteration when the input words have typical spellings. The accuracy of this system can be increased further by increasing the size of the training corpus and having plentiful of data covering maximum senses of all ambiguous words in the target script.

Conclusion

For the first time a high accuracy Gurmukhi-Shahmukhi transliteration system has been developed, which can accept data in any of the popular Punjabi font encoding. This will fulfil the demand from the users to develop such a system, where the Gurmukhi text in any font could be converted to Shahmukhi.

References

Lehal G.S. (2009), A Gurmukhi to Shahmukhi Transliteration System. In *Proceedings of ICON: 7th International Conference on Natural Language Procesing,* pages 167-173, Hyderabad, India.

Lehal, G. S. (2007). Design and Implementation of Punjabi Spell Checker, *International Journal of Systemics, Cybernetics and Informatics*, pages 70-75.

Malik, M.G.A. (2006). Punjabi Machine Transliteration. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL*, pages 1137-1144.

Saini, T. S., Lehal, G. S. and Kalra, V. S. (2008). Shahmukhi to Gurmukhi Transliteration System. In *Proceedings of 22nd international Conference on Computational Linguistics (Coling)*, pages 177-180, Manchester, UK.

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

Thede, S.M., Harper, M.P. (1999). A Second-Order Hidden Markov Model for Part-of-speech Tagging, In *Proceedings of the 37th annual meeting of the ACL on Computational Linguistics,* pages 175-182.

Davis, M., Whistler, K. (Eds.) (2010). Unicode Normalization Forms, *Technical Reports*, Unicode Standard Annex #15, Revision 33. Retrieved February 22, 2011, from http://www.unicode.org/reports/tr15/tr15-33.html