International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878, Volume-8 Issue-3, September 2019

An Efficient Romanization of Gurmukhi Punjabi Proper Nouns for Pattern Matching

Harjit Singh, Ashish Oberoi

Abstract*: A Romanization system is used to convert some text of a source script to the Roman script through word by word mapping. The phonological characteristics of the source word are not lost. Only writing script is changed, without any changes in the spoken language. This paper presents a rule based approach for Romanization of Gurmukhi script proper nouns. The aim is to develop a lightweight Romanization system, which may produce multiple possible results for the same input word. The algorithm uses a list of Gurmukhi script characters along with their equivalent character combinations in Roman script. Direct mapping of Gurmukhi script characters to their equivalent Roman script character combinations does not produce efficient results, so some rules are applied to get the correct mappings. The rules are basically to place or remove the letter 'a' in between the mapped consonants. Three different sets of rules are applied to get three different Romanized outputs. All these outputs are acceptable for information extraction using pattern matching. In Gurmukhi, some words are written differently than these are pronounced. To handle such words, these words or part of these words are stored in a database table. Along with these words their Romanized form is also stored in second column. The table is used to directly pick the Romanization from the table and use it for Romanization of these words. The result of this Romanization system is a set of possible words that can be generated from the source script word. It enables an application to pattern match those output words with some text or database to get the required information.

***Index Terms*: Gurmukhi Punjabi, Natural Language Processing, Rule Based, Romanization.**

I. INTRODUCTION

A Romanization system is used to convert some text of a source script to the Roman script through word by word mapping [1]. Only writing script is changed, without any changes in the spoken language. The phonological characteristics of the source word are not lost [2]. For example:

ਪੰਜਾਬ → Punjab
ਚੰਡੀਗੜ੍ਹ → Chandigarh
ਪਟਿਆਲਾ → Patiala
ਲੁਧਿਆਣਾ → Ludhiana

Manuscript published on 30 September 2019
* Correspondence Author
Harjit Singh*, APS Neighbourhood Campus, Punjabi University, Patiala, India. (Research Scholar at RIMT University)
Ashish Oberoi, School of Engineering, RIMT University, Mandi-Gobindgarh, India.

For this process, the source script word is first segmented into small units and then these small units are mapped to the Roman script. For example:

ਪਟਿਆਲਾ → ਪ ਟ ਿ ਅ ਾ ਲ ਾ → P A T I A L A

The resultant word need to be phonetically equivalent to the source word. In corpus based approaches, the source script word is not segmented and the equivalent Roman word is picked from the corpus [3].

Efficient transliteration of Named Entities of the source script to the target script improves the efficiency of Machine Translation [4]. Transliteration also plays an important role in Cross Language Information Retrieval and Information Extraction where some word(s) written in a particular script is/are searched in some different script text [5][6].

It is obvious that some phonetic symbols that are present in a source script may be missing in the target script [7]. Secondly, mostly the scripts are different in terms of their character sets and the way of writing [8]. For example, Roman script used to write English language has 26 characters, while Punjabi uses 41 characters. Some sounds of Punjabi are not phonetically present in the Roman alphabet. The way of writing scripts can also be different. For example, Arabic is written from right to left but English and Punjabi are written from left to right [9]. Thirdly, some Named Entity in the source text can be a mixture of two or more languages such as the word Bay-of-Bengal [10]. These types of words need to be transliterated very carefully. Lastly, a Named Entity can be transliterated to multiple variants in the target script with possibly all correct transliterations. For example, the name ਮਨਜੀਤ can be transliterated as MANJIT or MANJEET and both are correct transliterations. Both need to be considered for pattern matching.

II. MACHINE TRANLITERATION APPROACHES

Literature presents various approaches for transliteration depending upon the supported languages and methodologies used for the process. All these approaches can be divided into three categories i.e. Grapheme Based Approaches, Phoneme Based Approaches and mixture of these two i.e. Hybrid Approaches [11]. Grapheme based approaches can further be Rule Based, HMM (Hidden Markov Model) Based, Statistical Based and Finite State Transducers [12]. Figure 1 shows these categories graphically. Since this paper presents a Rule based approach for transliteration, so it comes under Grapheme Base Approaches.

Grapheme Based Approaches

The basic, meaningful and grammatically correct unit used in a written language is called Grapheme [11]. These approaches work by mapping the source grapheme sequence to the target script directly.

Any phoneme level processes are ignored and the source graphemes are converted to the target script by character to character(s) mappings [12].

These approaches are also called direct approaches of machine transliteration [13]. These approaches can further be classified as:

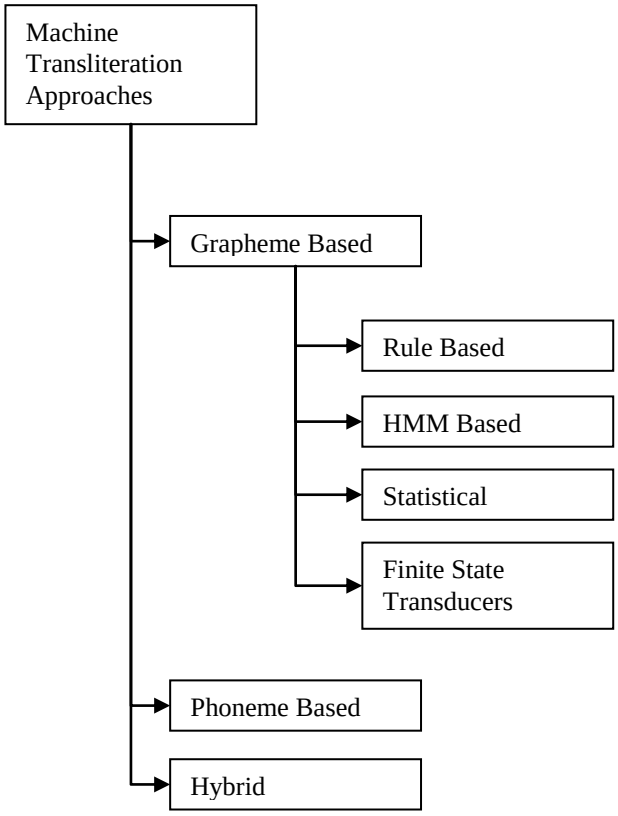

Figure 1: Machine Transliteration Approaches

***Rule Based Approach*:** In Rule based approach, a set of rules are defined which are processed by some algorithm(s). The system may also use some lexicon(s) to improve efficiency. The rules may be hardcoded in program(s) or may be stored in some database accessible to the program(s). The source text is processed by the program(s) and the defined rules are applied while the characters of source text are mapped to the target script. The rules are defined by human experts based on the syntactic, semantic and morphological information of both the languages. Pattern matching is applied on the source text to apply these rules and get the target text [14].

HMM Based Approach: HMM (Hidden Markov Model) is also a statistical model but different from other statistical models. These are frequently used for POS (Part-Of-Speech) Tagging a language text. It takes the source language text as a sequence $X=(X_1,X_2,X_3....X_n)$ and tries to predict the most suitable target language text as a sequence $Y=(Y_1,Y_2,Y_3....Y_n)$. The model uses a finite set of states having probability distributions. Transition from one state to another is controlled by the set of Transition Probabilities (P) and the output is controlled by the set of Output Probabilities (O). Figure 2 shows the states and probabilities of HMM based approach. HMM uses a tagged corpus and looks for all possible combinations of the source language text in the target language text and calculates probabilities of co-occurrence of the words. The model is also used for transliteration from the source script to the target script based on probabilities [15].

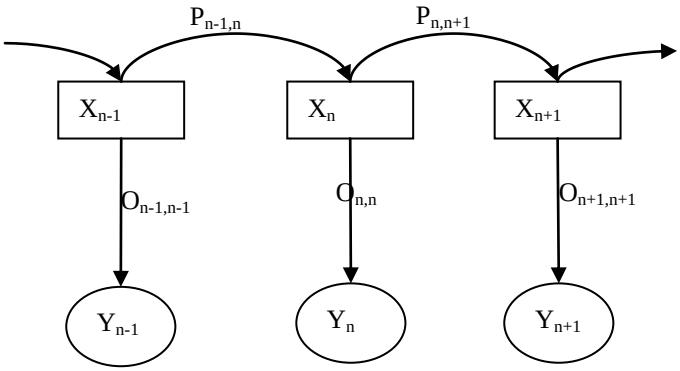

Figure 2: HMM Based Approach for Transliteration

Statistical Approach: Statistical approach is mostly used in translation instead of transliteration. In translation, bilingual parallel corpus is used to map the source language sentence to the target language sentence based on the probabilities of correct translation. A mathematical model is used to calculate probabilities of correctness of each sentence in the target language for a sentence in the source language. Then the sentence with the highest probability is chosen as the correct translation. The same model is used in transliteration with the variation that the bilingual corpus should be a transliterated corpus. The approach is efficient but totally dependent on the size of the corpus used in the process. More suitable sentences mean more accuracy [16].

Finite State Transducers: Finite State Transducers are popular models used in computational linguistics and pattern recognition. The concept follows the terminology of Turing Machine with using two memory tapes called input tape and output tape. The process maps among two sets of symbols. It defines a relation between the two sets. It reads a set of strings from the input tape and outputs a set of relations to the output tape. It provides output when it shifts from one state to another state. The states labeled with input/output symbols are connected to each other to form a network, in which transition between states is also labeled. The input text is matched with input labels to produce the output text through output labels [17].

Phoneme Based Approaches

The smallest units of sound are called phonemes. The spoken form of the word instead of the written form is used to map the source word into target script. The method is also called Pivot method. This method captures the pronunciation of the source word and uses that phonetic representation as an intermediate representation to form the word in the target script. The process takes two steps. In first step, the phoneme is generated from the grapheme of source word, and in second step the phoneme is used to generate the grapheme in target script i.e. the target word [18]. Figure 3 shows the steps of transliteration using Phoneme based approaches.

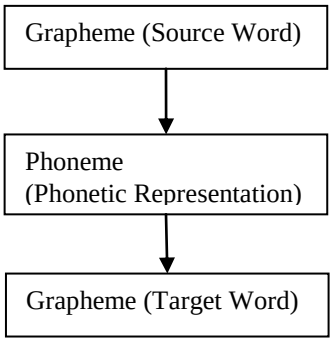

Figure 3: Phoneme Based Approach for Transliteration

So, in this approach the spellings of the source word are not important, rather, the pronunciation of the source word is important. This method makes sure the pronunciation similarity between source and target words. A word may contain one or more syllables. A syllable is a unit of pronunciation with one vowel sound and with/ without a consonant. For example, the word 'purple' has two syllables i.e. 'pur' & 'ple'. The source word is divided into syllables and these syllables are mapped to phonemes using some rules [19].

A. Hybrid Approaches

The Hybrid approaches mixes up both Grapheme based and Phoneme based approaches. Both the Grapheme and Phoneme of the source word are used to produce Grapheme in the target script. This approach can also be a combination of already discussed transliteration approaches i.e. the approach can combine Rule based and Statistical approach for transliteration [20].

III. RELATED WORK

S. Wan and C. M. Verspoor in their paper "Automatic English-Chinese name transliteration for development of multilingual resources", proposed a transliteration system for English to Chinese transliteration. The system was based on Phoneme based approach i.e. the spoken form of the English word is used to form the word in Chinese. The English word is mapped to corresponding phonetic representation which is then used to map to the Chinese grapheme. The system was developed for place names only, because the second step of phoneme to grapheme representation was very problematic. Chinese characters are monosyllabic, so the English word is divided into syllables through syllabification process. The output syllables are further divided in sub-syllables to map them into Chinese phonetic set. Some fixed set of rules were applied for both mappings [21].

B. J. Kang and K. S. Choi in their paper "Automatic Transliteration and Back-transliteration by Decision Tree Learning" presented a system for transliteration and back transliteration among English and Korean. The approach was based on decision tree learning [22].

J. H. Oh and K. S. Choi in their paper "An English-Korean transliteration model using pronunciation and contextual rules" presented English to Korean transliteration system. They developed an algorithm which they named as EPK algorithm. EPK is English-Phoneme-Korean. The algorithm maps the English grapheme to Phoneme and then Phoneme to Korean Grapheme [23].

C. J. Lee and J. S. Chang in their paper "Acquisition of English-Chinese transliterated word pairs from parallel-aligned texts using a statistical machine transliteration model" presented English to Chinese transliteration system. The system was based on Statistical Approach. A parallel corpus is used to map the source word to target word based on highest probability [24].

Verma, in his paper "A Roman-Gurmukhi Transliteration System" presented a Gurmukhi to Roman transliteration system named GTrans. The system used a Rule Based approach. Some rules were defined and these rules were applied by algorithm(s) for character by character mapping from Gurmukhi to Roman transliteration [25].

M. G. Malik in his paper "Punjabi machine transliteration" presented a Shahmukhi to Gurmukhi transliteration system based on rule based approach. One interesting thing is that both the scripts are already being used to write Punjabi language, so the pronunciation is preserved in transliteration. Both scripts use different characters sets. Transliteration rules were generated for character to character mapping among Shahmukhi and Gurmukhi [26].

H. Surana and A. K. Singh in their paper "A More Discerning and Adaptable Multilingual Transliteration Mechanism for Indian Languages" proposed a transliteration system for Hindi and Telugu [27].

IV. METHODOLOGY

This paper presents the Romanization system for Gurmukhi Proper Nouns. The system uses Rule Based approach to convert Proper Nouns of Gurmukhi into equivalent Proper Nouns in English. The system uses character mappings shown in Table 1.

Table 1: Gurmukhi to Roman Character Mapping

Punjabi Character	English Equivalent	Consonant(c) / Vowel (v)
ਓ	o	v
ੳ	-	-
ਅ	a	v
ੲ	-	-
ਸ	s	c
ਹ	h	c
ਕ	k	c
ਖ	kh	c
ਗ	g	c
ਘ	gh	c
ਙ	ng	c
ਚ	ch	c
ਛ	chh	c
ਜ	j	c
ਝ	jh	c
ਞ	nj	c
ਟ	t	c

ਠ	th	C
ਡ	d	c
ਢ	dh	c
ਣ	n	c
ਤ	t	c
ਥ	th	c
ਦ	d	c
ਧ	dh	c
ਨ	n	c
ਪ	p	c
ਫ	f	c
ਬ	b	c
ਭ	bh	c
ਮ	m	c
ਯ	y	c
ਰ	r	c
ਲ	l	c
ਵ	v,w	c
ੜ	r,rh	c
ਸ਼	sh	c
ਖ਼	kh	c
ਗ਼	gh	c
ਜ਼	z	c
ਫ਼	f	c
ਲ਼	l	c
ੇ	e	v
ਾ	a	v
ੈ	e	v
ਿ	i	v
ੀ	i,ee	v
ੋ	o	v
ੌ	au	v
ੁ	u	v
ੂ	u,oo	v
ਂ	n	c
ੰ	n	c
ੱ	double	c

The algorithm for transliteration uses this mapping list for mapping of Gurmukhi characters to equivalent English characters. The list is stored in the database with first column for Gurmukhi character, second column for English equivalent character(s) and third column specifies whether the English character is consonant or vowel. For the consonant, c is stored in the third column and for a vowel v is stored in this column. To make the process simple and lightweight for the motive of reducing the processing time, the algorithm takes following steps:

Step 1: The database table containing the above list is loaded into a two dimensional array. It has two advantages. First, using the array the list is loaded in main memory which fastens the processing. Secondly, while transliterating, the word need to be traversed character by character forward and backward, so array is most suitable structure for this task.

Step 2: All the Gurmukhi characters are directly mapped to equivalent English characters by the algorithm. The list stored in the two dimensional array is used for this mapping. Since the list is available in main memory, the system takes very less time to generate the word using equivalent English character(s).

For example, the Gurmukhi Proper Noun 'ਪਟਿਆਲਾ' is character mapped directly to equivalent English characters as per the list above as:

ਪ ਟ ਿ ਅ ਾ ਲ ਾ → P T I A A L A

But, as seen in this example, the resultant word is not a proper transliteration of original Gurmukhi word. The next step will make improvements to this word.

Step 3. To improve the results, the letter 'a' of English need to be placed/removed in between consonants wherever required. The placement/removal of 'a' requires some rules which are applied by the algorithm to successfully generate up to three possible romanizations using separate set of rules. Figure 4 shows the flowchart represenation of steps used by this algorithm.

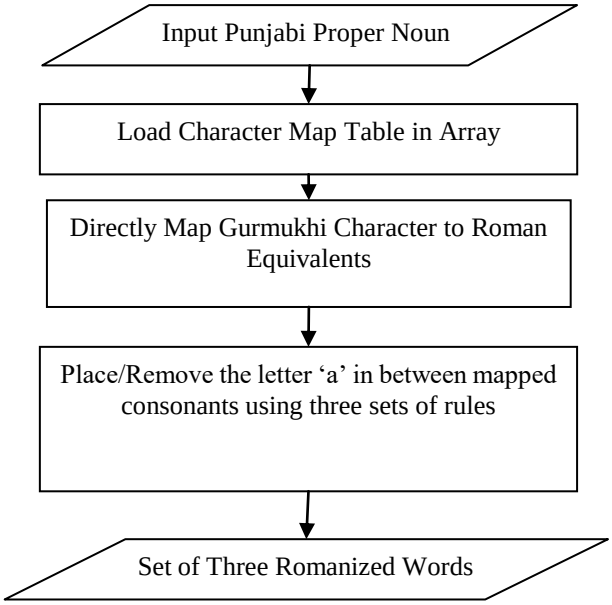

Figure 4: Flowchart of Algorithm used for Romanization

The results of transliteration of some words from the tests are given in the table 2. The most appropriate transliteration is shown in Bold in the table.

The table shows a proper noun ਹੁਸ਼ਿਆਰਪੁਰ that is not Romanized properly because it is pronounced differently than it is written.

Table 2: Romanization Results

Gurmukhi Proper Noun	Transliteration 1	Transliteration 2	Transliteration 3
ਮਨਜੀਤ-ਸਿੰਘ	**manjit-singh, manjeet-singh**	**manjit-singh, manjeet-singh**	manajit-singh, manajeet-singh
ਚੰਡੀਗੜ੍ਹ	**chandigarh**	chandigrh	chandigrh
ਰੋਪੜ	**ropar**	ropr	ropr
ਰਾਜਪੁਰਾ	rajapura	**rajpura**	**rajpura**
ਪਟਿਆਲਾ	**patiala**	**patiala**	**patiala**
ਖੰਨਾ	**khanna**	**khanna**	**khanna**
ਗੋਬਿੰਦਗੜ੍ਹ	gobindagrh	**gobindgarh**	**gobindgarh**
ਲੁਧਿਆਣਾ	**ludhiana**	**ludhiana**	**ludhiana**
ਸੰਗਰੂਰ	sangarur, sangroor	**sangrur,** sangroor	**sangrur,** sangroor
ਅਮ੍ਰਿਤਸਰ	amritasr	**amritsar**	**amritsar**
ਬਰਨਾਲਾ	**barnala**	**barnala**	**baranala**
ਬਠਿੰਡਾ	**bathinda**	**bathinda**	**bathinda**
ਫਰੀਦਕੋਟ	faridakot	**faridkot**	**faridkot**
ਫਤਹਿਗੜ੍ਹ-ਸਾਹਿਬ	fatehagrh-sahib	**fatehgarh-sahib**	**fatehgarh-sahib**
ਫਾਜ਼ਿਲਕਾ	fazilaka	**fazilka**	**fazilka**
ਫਿਰੋਜ਼ਪੁਰ	firozapur	**firozpur**	**firozpur**
ਗੁਰਦਾਸਪੁਰ	guradasapur	**gurdaspur**	**gurdaspur**
ਹੁਸ਼ਿਆਰਪੁਰ	hushiarapur	hushiarpur	hushiarpur
ਜਲੰਧਰ	**jullundhar**	**jullundhar**	**jullundhar**
ਕਪੂਰਥਲਾ	kapurathla, kapoorathla	**kapurthala,** kapoorthala	**kapurthala,** kapoorthala
ਮਾਨਸਾ	manasa	**mansa**	**mansa**
ਮੋਗਾ	**moga**	**moga**	**moga**
ਮੁਕਤਸਰ	**mukatsar**	muktasr	muktasar
ਨਵਾਂਸ਼ਹਿਰ	navanshehar, **nawanshehar**	navanshehr, nawanshehr	navanshehr, nawanshehr
ਸ਼ਹੀਦ-ਭਗਤ-ਸਿੰਘ-ਨਗਰ	shahid-bhagt-singh-nagr	shahid-bhagt-singh-nagr	**shahid-bhagat-singh-nagar**
ਪਠਾਣਕੋਟ	pathanakot	**pathankot**	**pathankot**
ਰੂਪਨਗਰ	rupangar, roopangar	rupnagr, roopnagr	**rupnagar,** roopnagar
ਸਾਹਿਬਜ਼ਾਦਾ-ਅਜੀਤ-ਸਿੰਘ-ਨਗਰ	sahibzada-ajit-singh-nagr	sahibzada-ajit-singh-nagr	**sahibzada-ajit-singh-nagar**
ਮੋਹਾਲੀ	**mohali**	**mohali**	**mohali**
ਤਰਨ-ਤਾਰਨ	**tarn-taran**	tarn-tarn	taran-tarn

Some Proper Nouns sounds differently than actually they are spelled in Punjabi. For example: the word 'ਫਤਹਿਗੜ੍ਹ-ਸਾਹਿਬ' is pronounced as 'ਫਤੇਹਗੜ੍ਹ-ਸਾਹਿਬ'. These type of words create problems while transliterating, because the English equivalent of this word is 'Fatehgarh-Sahib' which is a transliteration of 'ਫਤੇਹਗੜ੍ਹ-ਸਾਹਿਬ' not of 'ਫਤਹਿਗੜ੍ਹ-ਸਾਹਿਬ' as per the character equivalent list used by the algorithm. Some proper names were written in Roman differently before independence such as ਜਲੰਧਰ was written as Jullundhar, which does not match phonologically. The word ਪੰਜਾਬ is spelled in English as Punjab, but sometimes the word Panjab is used instead of Punjab e.g. in Panjab University Chandigarh. So, in information extraction we may need to match both versions through pattern matching. To solve the problem, such tokens are stored in the database with their equivalent English transliterations to reduce the number of rules and fasten the processing. For example, Table 3 shows some tokens, which may be the whole word or a part of some word that are pronounced differently than these are spelled:

Table 3: Some Punjabi Tokens pronounced differently

Punjabi Token	Equivalent English Transliteration
ਫਤਹਿ	fateh
ਸ਼ਹਿਰ	shehar
ਪੰਜਾਬ	punjab

V. IMPLEMENTATION AND TESTING

The above methodology is implemented in C#.NET and SQL Server as backend database using Visual Studio.NET 2010. Figure 5 shows the implementation interface of Romanization System for Gurmukhi Proper Nouns.

To automate the word input for testing, the data is stored in a Unicode text file with one noun per line and that text file is read line by line when the button 15? (shown in Figure 5) is pressed. The button displays the currect word number as a label.

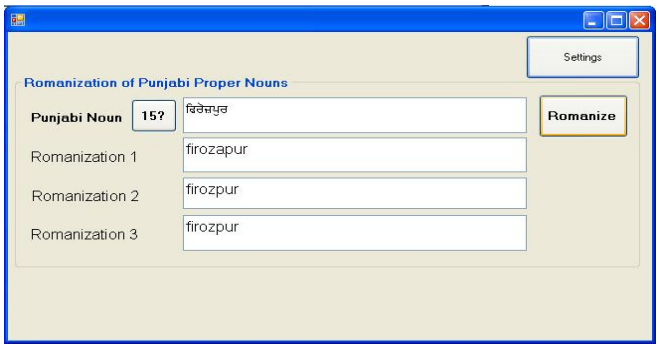

Figure 5: Implementation Interface of Romanization System

The implementation is tested using names of districts of Punjab (22), names of popular cities of Punjab (54) and 500 Punjabi names taken from University Rolls. A total of 576 proper nouns were used for testing the implementation. The system produces three results of Romanization to cover each possibility of spellings. For example, the name of a person ਕੁਲਜੀਤ. One person named ਕੁਲਜੀਤ may spell his name in Roman as KULJEET, while another person named ਕੁਲਜੀਤ may spell his name as KULJIT, both are correct. Similarly ਕਰਮਜੀਤ may be spelled in Roman as KARAMJIT, KARMJIT or KARMJEET. In pattern matching, all possibilities need to be searched, so these variations in output are correct for pattern matching and should be included in the result set. But some proper names are not romanized as they are popularly written in Roman. For example, district ਹੁਸ਼ਿਆਰਪੁਰ is not romanized properly as shown in Table 2. This problem can be easily solved by adding more such type of proper nouns with their English equivalents to the database table as shown in Table 3. The test results are shown in Table 4. Figure 6 graphically represents the test results.

Table 4: Test Results

Test Set: Test-1 Test-2 Test-3	Total Number of proper nouns	No. of proper nouns Romanized properly	Percentage of Accuracy
Names Districts of Punjab	22	21	95.4%

Names of popular cities	54	51	94.4%
Punjabi names taken from University Rolls	500	488	97.6%
		Average Accuracy	95.8%

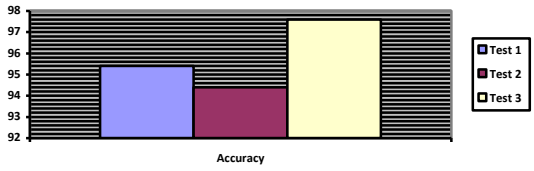

Figure 6: Graph of Accuracy Percentage

VI. CONCLUSION

This paper presented a rule based approach for Romanization of Gurmukhi script proper nouns. It is a lightweight Romanization system, which may produce multiple possible transliterated results for the same source script word. The algorithm used a list of Gurmukhi script characters along with their equivalent character combinations in Roman script. Direct mapping of Gurmukhi script characters to their equivalent Roman script character combinations does not produce efficient results, so some rules are applied to get correct mappings. Rules are used to place or remove the letter 'a' in between the mapped consonants. Differently pronounced words are handled by picking their Romanized form directly from the database table. The output of this Romanization system is a set of possible words that can be generated from the source script word. It enables an application to pattern match those output words with some text or database to get the required information (information extraction). The algorithm used to implement this system provided 95.8% accuracy when tested on various types of Gurmukhi Punjabi proper nouns.

REFERENCES

1. Antony P J,Dr. Soman K P, "Machine Transliteration for Indian Languages: A Literature Survey", International Journal of Scientific & Engineering Research, Volume 2, Issue 12, December-2011
2. Arvind Iyengar, "Romanisation of Indian languages: a diachronic analysis of its failure", In 31st South Asian Language Analysis Roundtable Conference, Lancaster University, 2015
3. R. Srivastava and R. A. Bhat. "Transliteration systems across indian languages using parallel corpora",In Proceedings of the 27th Pacific Asia Conference on Language, Information, and Computation (PACLIC 27), pages 390–398. Department of English, National Chengchi University,2013.
4. Ulf Hermjakob, Kevin Knight, Hal Daum´e III, "Name Translation in Statistical Machine Translation Learning When to Transliterate", Proceedings of ACL-08: HLT, pages 389–397, Columbus, Ohio, USA, June 2008.
5. Fujii, Atsushi, Tetsuya Ishikawa, "Cross-Language Information Retrieval for Technical Documents", Proceedings of the Joint ACL SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora. 29-37. 1999.
6. Céline Paganelli, Evelyne Mounier, "Information retrieval in technical documents: from the user's query to the information-unit tagging", SIGDOC '03 Proceedings of the 21st annual international conference on Documentation Pages 133 - 139, San Francisco, CA, USA — October 12 - 15, 2003
7. Gurpreet Singh Josan, Jagroop Kaur, "PUNJABI TO HINDI STATISTICAL MACHINE TRANSLITERATION", International Journal of Information Technology and Knowledge Management, Volume 4, No. 2, pp. 459-463, July-December 2011
8. S. Karimi, F. Scholer, and A. Turpin, "Machine transliteration survey," ACM Computing Surveys (CSUR),vol. 43, p. 17, 2011.
9. Bushra J. Tafseer A.,"Hindi to Urdu Conversion: Beyond Simple Transliteration", Proceedings of the Conference on Language & Technology, Lahore 24-31, 2009
10. S Thara, Prabaharan Poornachandran, "Code-Mixing: A Brief Survey", International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2018
11. Kamaljeet Kaur, Parminder Singh, "Review of Machine Transliteration Techniques", International Journal of Computer Applications (0975 – 8887), Volume 107 – No. 20, December 2014.
12. Kanwaljit Kaur, "Machine transliteration: A Review of Literature", International Journal of Engineering Trends and Technology (IJETT) – Volume 37 Number 6 - July 2016.
13. Jong-Hoon Oh, Key-Sun Choi, Hitoshi Isahara, "A machine transliteration model based on correspondence between graphemes and phonemes", ACM Transactions on Asian Language Information Processing (TALIP), Volume 5 Issue 3, Pages 185-208, September 2006.
14. Deepti Bhalla, Nisheeth Joshi, Iti Mathur, "Rule Based Transliteration Scheme for English to Punjabi", International Journal on Natural Language Computing (IJNLC) Vol. 2, No.2, April 2013.
15. Asif Ekbal, Sudip Kumar Naskar, Sivaji Bandyopadhyay, "Named Entity Recognition and transliteration in Bengali", Lingvisticæ Investigationes, Volume 30, Issue 1, p. 95 - 114, Jan 2007.
16. Nasreen AbdulJaleel, Leah S. Larkey,"Statistical transliteration for english-arabic cross language information retrieval", CIKM '03 Proceedings of the twelfth international conference on Information and knowledge management, Pages 139-146, New Orleans, LA, USA — November 03 - 08, 2003
17. Mehryar Mohri,"Finite-state transducers in language and speech processing", Computational Linguistics, Pages 269-311, Volume 23, Issue 2, June 1997
18. Wei GaoKam-Fai, WongWai Lam,"Phoneme-Based Transliteration of Foreign Names for OOV Problem", Natural Language Processing – IJCNLP 2004, pp 110-119, 2004.
19. Sujith Ravi, Kevin Knight,"Learning Phoneme Mappings for Transliteration without Parallel Data", Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 37–45, Boulder, Colorado, June 2009.
20. Abbas Malik, Laurent Besacier, Christian Boitet, Pushpak Bhattacharyya, "A hybrid model for Urdu Hindi transliteration", NEWS '09 Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration, Pages 177-185, Suntec, Singapore, August 07, 2009.
21. S. Wan, C. M. Verspoor, "Automatic English-Chinese name transliteration for development of multilingual resources," in Proceedings of the 17th international conference on Computational linguistics-Volume 2, pp. 1352-1356, 1998
22. B. J. Kang, K. S. Choi, "Automatic Transliteration and Back-transliteration by Decision Tree Learning," in LREC, 2000.
23. J. H. Oh and K. S. Choi, "An English-Korean transliteration model using pronunciation and contextual rules," in Proceedings of the 19th international conference on Computational linguistics-Volume 1, pp. 1-7, 2002.
24. C. J. Lee, J. S. Chang, "Acquisition of English-Chinese transliterated word pairs from parallel-aligned texts using a statistical machine transliteration model," in Proceedings of the HLT-NAACL 2003 Workshop on Building and using parallel texts: data driven machine translation and beyond-Volume 3, pp. 96-103, 2003.
25. Verma, "A Roman-Gurmukhi Transliteration System", Proceeding of the Department of Computer Science, Punjabi University, Patiala, 2006.
26. M. G. Malik, "Punjabi machine transliteration," in Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics, pp. 1137-1144, 2006.
27. H. Surana, A. K. Singh, "A More Discerning and Adaptable Multilingual Transliteration Mechanism for Indian Languages," in IJCNLP, pp. 64-71, 2008.

AUTHORS PROFILE

Mr. Harjit Singh received the MCA (Master in Computer Applications) degree from IGNOU (Indira Gandhi National Open University), New Delhi, India. Alongwith pursuing Post Graduation he worked as a Web Developer. He acquired M.Phil.(CS) degree along with working as a Web professional. Presently he is working as Assistant Professor (Senior Scale) in Computer Science at Akali Phoola Singh Neighbourhood Campus under Punjabi University, Patiala (India) and is pursuing his Ph.D. degree. His current research interests include Natural Language Processing, Machine Translation, Artificial Intelligence.

Dr. Ashish Obroi is working as Professor in Computer Science and Engineering at School of Engineering, RIMT University, Mandi Gobindgarh, Punjab, India. He completed his Ph.D. in Computer Science and Engineering from Maharishi Markandeshwar University, Mullana, India. He is skilled and expertise in Image Processing, Medical Imaging, Image Segmentation, Diagnostic Imaging and Image Reconstruction.