Adaptive Transliteration Based on Cross-Script Trie Generation: A Case of Roman-Urdu

Usman Afzal, Dr. Naveed Iqbal Rao and Ahmad Muqeem Sheri
Image Processing Center, Military College of Signals,
National University of Sciences and Technology, Rawalpindi, Pakistan
{usman.afzal, naveedi}@mcs.edu.pk, muqeem.sheri@gmail.com

Abstract

In past few years many tools and techniques have been developed to support language localization. Keyboard layouts and IMEs have been developed and standardized to support local language scripts. But people commonly use Romanized versions of their local languages for Internet and mobile chat. Like many other Asian languages, Urdu is also used in Romanized form (Roman-Urdu) which is very popular among Urdu speakers. There is no single globally accepted standard to write Roman-Urdu, so people write Roman-alphabet spellings by intuition with respect to their language experience. Therefore, retrieving back the original Urdu word from a given Roman-spellings (reverse-transliteration) is a challenging problem.

In this paper, we propose an adaptive cross-script trie model which solves the reverse-transliteration problem effectively. The model consists of three layers: a) pre-processing, b) cross-script mapping, and c) trie generation. Pre-processing layer simplify the input word for the next layer. Cross-script mapping layer is the core of the model which performs mapping and transformation across the scripts. This layer is totally based on the analysis of vowel and consonant mappings which is another contribution we present in this paper. Our model returns all possible equivalent Urdu words as a trie for any given Roman-Urdu word. Trie generation layer uses trie-pruning to remove false branches. Experimental results have proved the significance and effectiveness of our model against variations of roman-spellings in test data.

1. Introduction

Writing a language in its customary script is a basic need of speakers of any language. With advancement in technology and language computing, many techniques and tools have been developed to support language localization, especially for Asian languages [1], [2]. Standardization of keyboard layouts and development of different Input Method Editors (IMEs) have been the primary focus of language localization efforts. The purpose of an IME is to allow users to input characters of their local languages using standard Latin keyboards. Microsoft Windows XP provides many IMEs for East Asian languages like Chinese, Japanese, and Korean [3]. Pronunciation-based IMEs are very helpful for users who are less familiar with language specific keyboards and use transliterated versions of their languages using Roman alphabets.

Like many other Asian languages (Arabic, Persian, and Hindi) Urdu is also written using Roman alphabets, and is named as Roman-Urdu. Although there are different interfaces and layouts available to input Urdu characters, majority of the language speakers prefer to use Roman-Urdu, especially for Internet and mobile chat. The obvious reason is the unfamiliarity and less use of Urdu keyboards.

Transliteration is a transformation of text from one script to another, usually based on phonetic equivalences. Roman alphabets are most commonly used for transliteration of languages which have non-Latin scripts. A transliteration scheme defines unambiguous one-to-one mapping of characters across the scripts. Urdu language has many different transliteration schemes [4]-[11] and, therefore, no single set standard is followed by the speakers to write Roman-Urdu. Roman-alphabet spellings are highly dependent on accent and educational level, so everyone writes roman spellings of Urdu words by intuition. As a result, there exists multiple spelling combinations equivalent to one Urdu word, as shown in Figure 2(a). This diversity of roman spellings makes retrieval of the original word (technically known as *Reverse transliteration*) very difficult. Transliteration is not trivial to automate, but reverse transliteration is even more challenging problem.

چ	ج	ث	ٹ	ت	پ	ب	ا
ch	j	c	T	t	p	b	a
ز	ڑ	ر	ذ	ڈ	د	خ	ح
z	R	r	Z	D	d	Kh	H
ع	ظ	ط	ض	ص	ش	س	ژ
a'	z^	t^	Z^	S	sh	s	zh
ن	م	ل	گ	ک	ق	ف	غ
n	m	l	g	k	q	f	Gh
		ں	ے	ی	ھ	ہ	و
		N	Y	y	h~	h	v,w

(a) Transliteration scheme

(1) ضبط Z^abt^
{zəbt, discipline}[1]

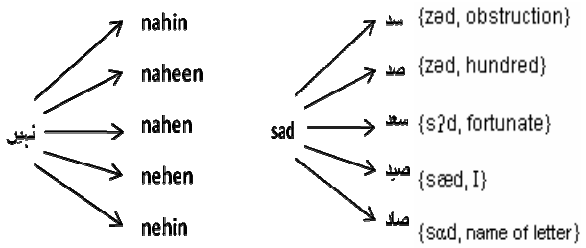
(2) صبر Sabar
{səbər, patience}

(b) Example words

Figure 1: Transliteration scheme with example

The number of Urdu alphabets is greater than the number of Roman alphabets, so every transliteration scheme uses some special symbols and capitalization of Roman-alphabets to unambiguously map all Urdu characters. But language speakers do not prefer to use special symbols or capitalization while writing Roman-Urdu in their casual conversation. An example of a transliteration scheme is shown in Figure 1. South Asian languages especially Urdu has given less attention in automatic NLP [14]. Urdu transliteration systems found in literature are single-scheme systems and are very specialized. A better transliteration system is available for Persian language which is based on statistical language modeling technique. It does not use special symbols and it also handles diversity of roman spellings to some extent [13].

Interestingly, the problem of Urdu reverse transliteration is two-fold. Firstly, one Urdu word can have more than one Roman-Urdu spellings and secondly, one Roman spellings can correspond to more than one word in Urdu as shown in Figure 2. An ideal solution to this problem is to have a parallel corpus of Roman-Urdu, listing all possible roman equivalents for all Urdu words. But, unfortunately, no such corpus is available to date. The available corpus is in Urdu script in Unicode format [15]. Statistical and N-gram based language modeling techniques are effective for many applications like spellchecking, auto-correct, and information retrieval [16]-[20]. These techniques



(a) Urdu to Roman script (b) Roman to Urdu script

Figure 2: Transliteration two-fold mapping.

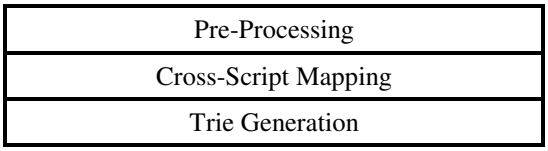


Figure 3: Cross-script trie generation model

cannot be employed for reverse transliteration problem of Roman-Urdu unless a parallel corpus is available.

An adaptive cross-script trie generation model is proposed in this paper (explained in section 3) and a transliteration system for Roman-Urdu was developed which gave favorable results. Our model is a dictionary based solution which utilizes the available corpus in Urdu script [21]. We structure our model as three layer architecture, shown in Figure 3. Cross-script mapping layer (explained in section 3.2) performs the core functionality in our model. Mapping is performed letter-by-letter which returns all possible Urdu characters for the given Roman alphabet(s). It is totally based on the analysis of vowel and consonant mappings (section 2) across the two scripts: *Urdu script* and *Roman script*. Based on these mappings, a cross-script trie is generated for the given roman word at the last layer. The leaf nodes of this trie provide a list of possible Urdu words equivalent to the given Roman-Urdu word.

The contributions we have presented in this paper are:

- A precise analysis of cross-script vowel and consonant mappings.
- A cross-script trie generation model is proposed which supports two major applications: (a) adaptive transliteration of Roman-Urdu, and (b) cross-script phonetic search.
- An implementation of the proposed model as a simple application of reverse transliteration for causal Roman-Urdu. A screen short is given in Figure 4.

[1] Examples of Urdu-script are quoted along with their IPA equivalent and English meaning in the form {IPA, Meaning}

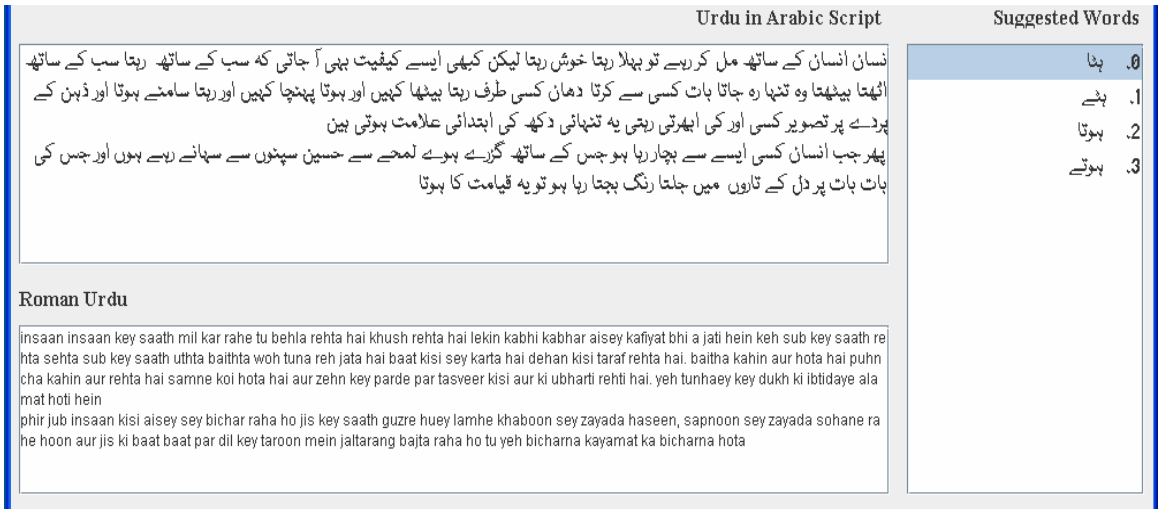


Figure 4: Screenshot of application developed for reverse transliteration of Roman-Urdu. (*Note:* User can select his desired word from 'Suggested Words List' by pressing Ctrl+Num keys)

2. Vowel and consonant mappings

Urdu has 29 basic and 4 secondary characters. Combination of these characters results in a rich inventory of 44 consonants, 15 long vowels and 3 short vowels [22]. In this section we present a precise discussion of vowel and consonant mappings across the Roman and Urdu scripts. This discussion is based on a detailed analysis of many different transliteration schemes defined for Roman Urdu and also on the usage patterns of vowels and consonants in casual Roman-Urdu.

Vowels and consonants in Roman-Urdu usually map to more than one Urdu character. This one-to-many mapping create ambiguity in reverse transliteration which is the core problem of Urdu transliteration. Vowels and consonants map differently in Roman-Urdu and they both have their own relevant issues. So, we discuss these issues under separate subsections for simplicity. Vowel and consonant mappings are discussed in section 2.1 and section 2.2 respectively.

2.1. Vowel mappings

As mentioned earlier, Urdu has 15 long and 3 short vowels. To pronounce these vowels in Roman-Urdu, different people use different Roman-alphabets depending upon their accent and level of education. We identify two main issues in vowel mappings which are described in sections 2.1.1 and 2.1.2.

2.1.1. Shot vs. long vowels. Short vowels of Urdu language are pronounced and are normally written in Roman-Urdu, but they are not written in Urdu script. So, short vowels do not require any mappings from Roman alphabet to Urdu character. On the other hand, long vowels are written in Urdu script and, therefore, must be mapped to some Urdu character. In case of a strict transliteration scheme, each short and long vowel is designated with different Roman alphabet(s) to disambiguate the mappings along with the rules which apply according to positional context (initial, middle, or final) of the vowels. So there is no problem while retrieving Urdu script back as far as the rules of the particular scheme are followed. But in casual Roman-Urdu, which follows no standard, a single Roman alphabet is used for both short and long vowels and speakers of the language understand and distinguish these vowels by their context.

For example, according to transliteration scheme shown in Figure 1, صاد {sɑd, name of letter}can only be written as '*saad*' as a double 'aa' is assigned to the middle and final position of character 'ا' (initial position of this character is assigned a single 'a'). In casual Roman-Urdu on the other hand, صاد {sɑd, name of letter} can be written as '*sad*' or '*saad*'. But '*sad*' can also refer to the word سد {səd, hundred}. So, when converting back '*sad*' to Urdu script, the vowel '*a*' creates ambiguity and makes it very difficult to determine either it's a short vowel and should be mapped to سد {səd, obstruction} or it's a long vowel and should be mapped to صاد {sɑd, name of letter}.

2.1.2. Overlap of long vowels mapping. Urdu long vowels are represented in Roman-Urdu by many

different Roman alphabets. Again it depends on the user's intuition while writing Roman spellings. So, different users can use the same Roman alphabet for representing different long vowels. This creates an overlap of Roman alphabets for a single long vowel. For example, صید {sæd, prey} is written either as '*sad*' or '*sed*'. Here '*a*' and '*e*' both can be mapped to the same Urdu vowel, represented as 'ی' in this example.

2.2. Consonant mappings

Consonants are easier to map as compared to vowels. Each consonant in Roman-Urdu must be mapped to some Urdu character. There are two types of consonant mappings:

a) *One-to-one consonant mapping* is used for consonants which have unique sounds. For example, b for ب, p for پ, and l for ل.
b) *One-to-many consonant mapping* is used for groups of consonants which sound similar to each other. For example, t is mapped to all consonants in the group (ت, ٹ, and ط); and s is mapped to (ث, س, ص).

In general, we name all the roman alphabets which have one-to-many mapping as 'ambiguous characters'. So, all vowels and the type (b) consonants described above are ambiguous characters and they need special treatment in any reverse transliteration system.

3. Cross-script trie model

In this section, we present a detailed description of our proposed model (see Figure 5). We have discussed each layer of the model in a separate subsection. Our model is based on a knowledge base of Urdu words (94,216 words currently) available in the Urdu script. For simplicity, we are considering only un-spaced words but the model can be easily extended to spaced words by little change in the structure of the knowledge base.

Tries are ordered tree data structure, normally used to store and retrieve information from dictionaries. Tries store information as paths from root node to the leaf nodes and information is retrieved by traversing to the leaf nodes through the root node. Tries are also known as 'prefix trees' as the information stored in tries share common prefixes.

We have extracted 94,216 un-spaced Urdu words from the available corpus and we have stored them in a trie structure. As common prefixes of words are shared, our model can be extended to include cross-script word prediction and error correction features. An example of a simple trie build upon words کان {kɑn, ear}, کامل {kɑmɪl, perfect}, عمر {umər, age}, and علم {ɪlm, knowledge}, is shown in Figure 6.

There are two different tries in our model:

1) *a knowledge base trie* of Urdu words, which is generated once prior to the processing of the model
2) *a word-trie* which is generated for each input Roman-Urdu word.

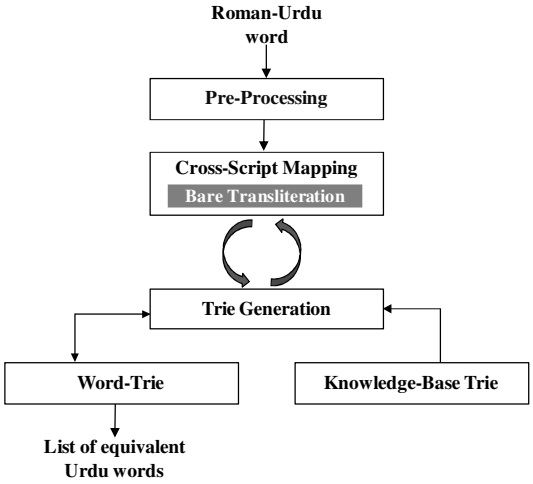


Figure 5: Detail view of cross-script trie generation model

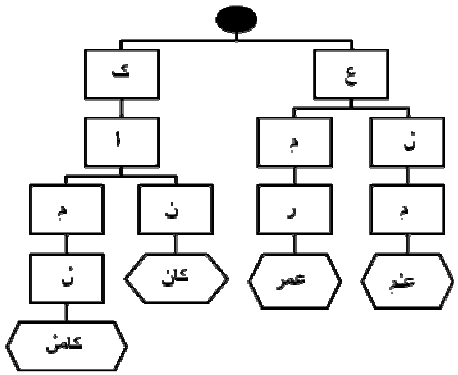


Figure 6: Example of trie consisting of words کان {kɑn, ear}, کامل {kɑmɪl, perfect}, عمر {umər, age}, and علم {ɪlm, knowledge}

3.1. Pre-processing

This is the first layer of the proposed model which gets a Roman-Urdu word as input and simplifies the word by performing some pre-processing steps. At the first step, the Roman word is converted into its

lowercase equivalent. This simplifies the implementation of the following layer of cross-script mapping explained in section 3.2.

To pronounce the effect of 'tashdeed' (which is used to strengthen a particular consonant in Urdu language) in a word, the consonant is normally written twice in Roman-Urdu. The pre-processing layer normalizes these double consonants into singles in the second step. For example, 'tamaddun' (تمدن) {təmədun, civilization} will be simplified as 'tamadun'. On the other hand, if there are any double vowels in Roman word, they are not normalized in this way because double vowels are not written due to 'tashdeed'; rather they are normally used to represent long vowels.

3.2. Cross-script mapping

The purpose of cross-script mapping (CSM) layer is to map the given Roman alphabet to corresponding possible Urdu character(s). This layer performs two important functions: **cross-script mapping** and **bare transliteration**. Bare transliteration is a sub-layer of CSM layer which is discussed in section 3.2.1.

Cross-script mapping is based on vowel and consonant mapping analysis given in section 2. Mapping is performed letter-by-letter; one letter is taken at a time from the given Roman word along with its positional context. Based on the letter and its context, category of mapping (one-to-one or one-to-many) is determined as discussed in section 2 and finally the appropriate mapping is performed. Output of this mapping becomes the input of Bare Transliteration sub-layer; therefore, the output is encoded according to the transliteration scheme implemented by the sub-layer. This is illustrated in Figure 7.

3.2.1. Bare transliteration. This sub-layer performs the actual transformation across the scripts. This layer behaves like a black-box which takes transliterated Urdu generated by the CSM layer and returns its equivalent Urdu script, as shown in Figure 8. We have separated the transformation from mapping and put it in a separate sub-layer. This gives the freedom to use existing implementations which actually perform only bare transliteration functionality. We have implemented the scheme proposed in [10] for our application, which is very phonetic in nature.

3.3. Trie generation

This layer works in parallel with the CSM layer and generates word-trie for the given Roman-Urdu word. Each time the CSM layer generates its output for the current Roman alphabet, it passes control to trie generation layer which receives a list of candidate Urdu characters. Each candidate character is temporarily added to word-trie at different nodes and verified from the knowledge-base trie for its validity. If this verification returns true, the addition of the character in the word-trie is made permanent. Otherwise, the complete path from root to this temporary node is declared as 'false path'. When the list of candidate characters is exhausted, trie is pruned by dropping all the false paths. A complete example of trie generation is illustrated in Figure 9.

After the entire Roman word is mapped, word-trie is completed, providing a list of suggested Urdu words at leaf nodes which are equivalent to the given Roman word. Trie-pruning at each step improves the performance and helps avoiding the wrong suggestions, to ensure maximum accuracy of the model.

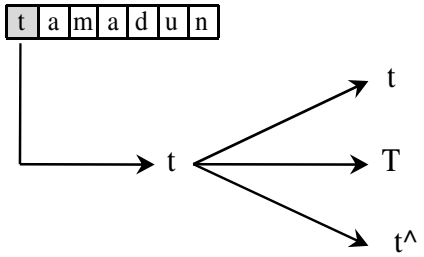


Figure 7: Output of mapping according to bare transliteration scheme

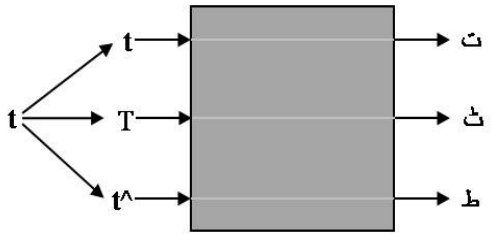


Figure 8: Bare transliteration black-box

4. Results and discussion

We have implemented our model in a simple application developed in Java programming language with Microsoft Access as the backend database. IBM provides different code packages to support applications dealing with Unicode under the project International Components for Unicode (ICU) [23], [24]. It also provides packages which support bare transliteration and we have used this package for our application.

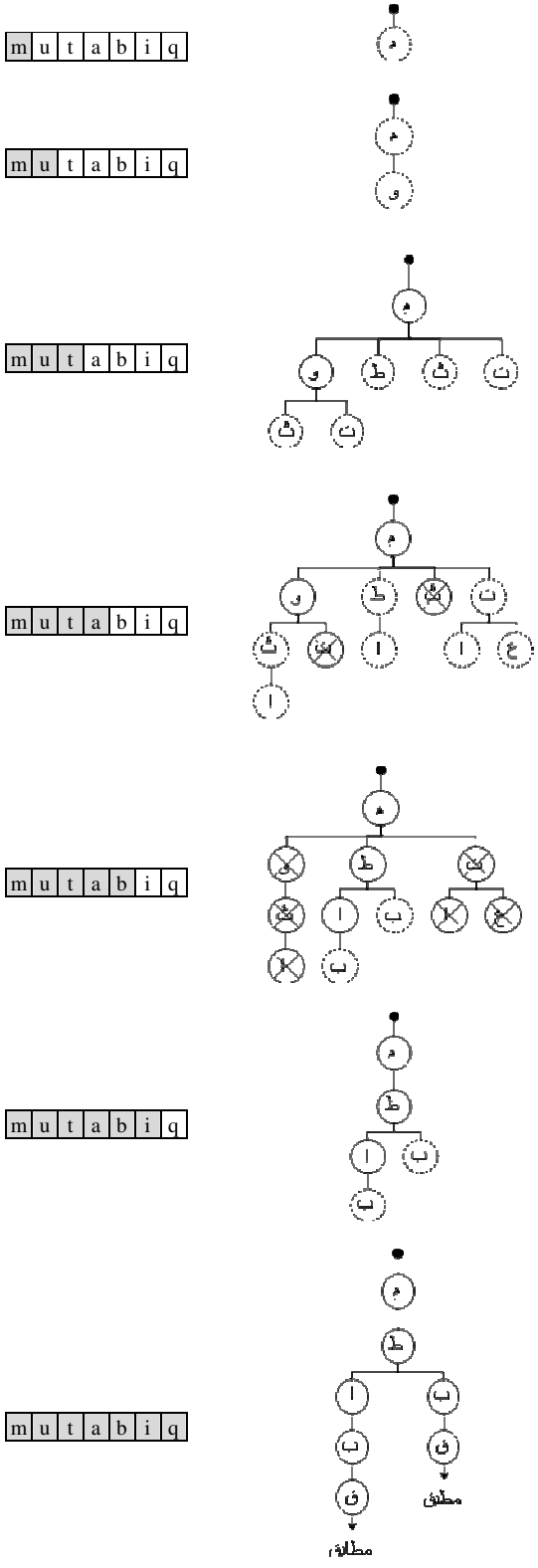


Figure 9: Cross-script trie generation

Our knowledge base covers 94,216 un-spaced words which we have extracted from a corpus of Urdu words collected from different sources [21]. To test the performance and accuracy of our model, we have randomly selected four poems and three text paragraphs written in Roman-Urdu from different Roman-Urdu websites available on the Internet[25]-[27]. Poems give a success rate[2] of 84.3% on average and paragraphs give a success rate of 85% on average. Results for poems and paragraphs are shown in Figure 9 and Figure 10 respectively. Both graphs show very small variability in success rate which proves the consistency of our model. The overall accuracy of the model based on these tests is 84.5% on average and the error rate is 15.5%. Our application is demonstrated in Figure 4 where some words in Urdu script are missing due to the error rate.

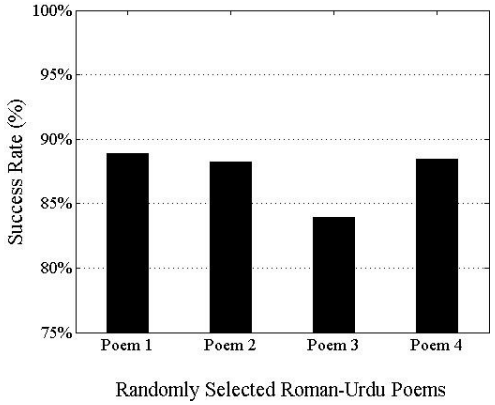


Figure 10: Success rate (%) for randomly selected poems

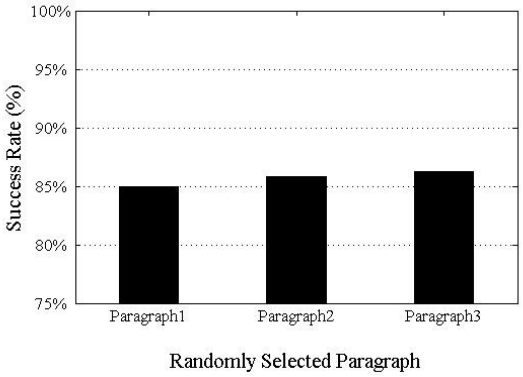


Figure 11: Success rate (%) for randomly selected Roman-Urdu paragraphs

[2] $SuccessRate\,(\%) = \frac{No.of\ input\ Roman\ words}{No.of\ output\ Urdu\ words} \times 100$

To test the effectiveness of our model, we have tested many different spellings used in Roman-Urdu for different Urdu words. We call different Roman spellings for same Urdu word as variants of a single word. For example تمدن can be written as '*tamadun*' or '*tmaddun*', so both Roman spellings are called variants of تمدن . We have randomly selected 15 words out of 94,216 words in our knowledge base, and split them into 5 groups of 3 words each. Every group is tested with different number of variants for each word in the group. Selected words along with their groups and number of variants are given in Table 1.

Table 1: Word groups and their success ratio

Group #	No. of variants		Words in the group	Success ratio
	Per word	Per group		
1	2	6	تصویر	6/6
			تفکرات	
			قریطہ	
2	4	12	کیفیت	11/12
			لجونتی	
			گلگھوٹو	
3	6	18	ابابیلیں	16/18
			نہیں	
			جراہیہ	
4	8	24	مسکراہٹیں	20/24
			نیلامبر	
			سرائی	
5	10	30	تسلیخی	23/30
			اعتراض	
			بھنبیریاں	

With increasing number of variants per word, the success rate is decreasing as shown by the curve in Figure 11. The minimum success rate we got is 76.67%.

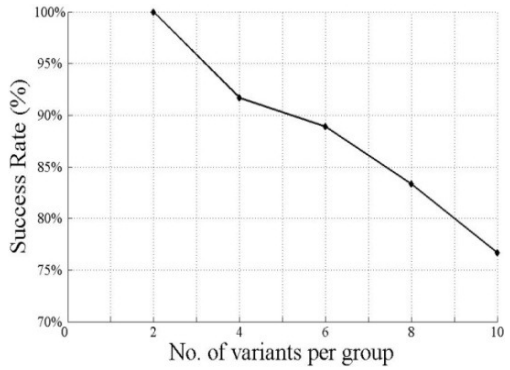


Figure 12: Roman word variants' success curve

5. Advantages and applications

The layered architecture of our proposed model has many advantages of extensibility and improvement. We have implemented the proposed model for Urdu language and achieved promising. The model can be extended to other languages like Arabic, Hindi, and Persian. The pre-processing steps in the model can be easily modified according to the language requirements without disturbing the functionality of other layers.

Performance and accuracy of the model is highly dependent on the vowel and consonant analysis given in section 2. The cross-script mapping (CSM) layer performs mapping and transformation based on this analysis. By just improving vowel & consonant analysis, overall performance of the model can be improved without any modification to other layers.

In addition to the adaptive transliteration, our model can be very useful for implementing a *phonetic search engine*. Although there are search engines available in Unicode to search Urdu resources. But, no cross-script phonetic search engine is available to date for Urdu, which can take Roman-Urdu as the search key and can hunt for resources of Urdu script. Moreover, the trie structure we used for our model has inherent benefits to add features like word prediction and error correction.

6. Conclusion

Reverse transliteration is a very supportive application for languages which have non-Latin scripts. Urdu language is one such case among many other Asian languages (Arabic, Persian, and Hindi). Due to less familiarity with Urdu keyboards, people use Roman-Urdu as their medium of communication over the Internet and mobile SMS. Due to unavailability of a single set standard to write Roman-Urdu, everyone

writes in its own way. To our knowledge, there is no tool available which converts casual Roman-Urdu into its equivalent Urdu script.

We have proposed a cross-script trie model which serves the purpose of reverse transliteration in an adaptive way. We have implemented our model which gives more than 75% accuracy with diverse nature of casual Roman-Urdu. Our model is applicable to other languages as well like Arabic, Persian, and Hindi.

We have tested our model for 94,216 un-spaced words and got an average success rate of 84.5%. The focus of our future work includes extension of the model for huge vocabulary containing spaced and joined words. We also intend to include loan words from other languages (e.g. English) which are commonly used by the speakers in their casual use of Roman-Urdu.

Our model is a dictionary based solution which has some inherent limitations; like, it cannot retrieve any word which is not in the dictionary. To cope with this limitation, we intend to improve the model so that it can update its knowledge base dynamically. Furthermore, the addition of an optimization layer can dramatically improve the usability of the model by sorting the list of output words based on some statistical model. We intend to add such a layer to our model in our future work.

7. References

[1] S. Hussain, N. Durrani and S. Gul, "*PAN Localization: Suvey of Language Computing in Asia*", CRULP NUCES, Pakistan, 2005.

[2] T. Rahman, "*Language Policy and Localization in Pakistan: Proposal for a Paradigmatic Shift*", Crossing the Digital Divide, SCALLA, 2004.

[3] Retrieved: http://www.microsoft.com/ globaldev/handson/user/IME_Paper.mspx

[4] F. Lodhi, *Urdu aur Farsi meN naqle Harfee* (**اردو اور فارسی میں نقل حرفی**), Natural Language Authority, Islamabad.

[5] A. Durrani , *Pakistani Urdu mazeed mubaaHis* (**پاکستانی اردو مزید مباحث**), Natural Language Authority, Islamabad.

[6] M. Humayoun, *Urdu Morphology, Orthography and Lexicon Extraction: Master's Thesis*, Chalmers University of Technology and Goteborg University, Sweden, 2006.

[7] Urdu Poetry Archive, Transliteration scheme for writing Urdu in English script. http://www.urdupoetry.com/ itrans.html

[8] Library of Congress, ALA-LC Romanization Tables: Transliteration Schemes for Non-Roman Scripts. http://www.loc.gov/catdir/cpso/roman.html

[9] UNGEGN (Working Group on Romanization Systems), "Report on the Current Status of United Nations Romanization Systems for Geographical Names", UNITED NATIONS GROUP OF EXPERTS ON GEOGRAPHICAL NAMES, 2003, Version 2.2.

[10] S. Raz., Roman Urdu (Research Paper), http://sarwarraz.com/tahqiqipage_ur.php?id=232&pageid=8 &title=rt-232.gif

[11] Rizvi S.R.M., Roman Urdu to Unicode Urdu text converter, http://www.geocities.com/syedrizwanm/UniCon vertFinal.html

[12] S. Khan, Z. Pervez, M. Mahmood, F. Mustafa and U. Hasan, "An Expert System Driven Approach to Generating Natural Language in Romanized Urdu from English Documents", *In proc IEEE INMIC*, 2003.

[13] Behnevis: easy farsi transliteration (pinglish) editor, Retrieved: http://www.behnvis.com/en/index.html

[14] W. Anwar, X. Wang and X.L. Wang, "A Survey of Automatic Urdu Language Processing", *Proceedings of the Fifth International Conference on Machine Learning and Cybernetics*, IEEE, Dalian, 2006.

[15] S. Hussain, "Resources for Urdu Language Processing", CRULP NUCES, Pakistan, 2008.

[16] P. Majumder, M. Mitra and B.B. Chaudhuri, "N-gram: a language independent approach to IR and NLP", Indian Statistical Institute, Kolkata,

[17] K. Kukich, "Techniques for Automatically Correcting Words in Text", ACM Computing Survey, 1992, pp.377-439.

[18] J. Goodman, "The State of the Art in Language Modeling", *Microsoft Research, Speech Technology Group*

[19] R. Rosenfeld, "Two Decades of Statistical Language: Modeling Where Do We Go From Here?", In proc *IEEE, 2004*.

[20] J. Allan, "Challenges in Information Retrieval and Language Modeling: Report of a Workshop held at the Center for Intelligent Information Retrieval", University of Massachusetts, Amherst, 2002.

[21] Urdu word list, CRULP NUCES, http://www.crulp.org/ software/ling_resources/wordlist.htm

[22] S. Hussain, "Letter-to-Sound Conversion for Urdu Text-to-Speech System", CRULP NUCES, 2004

[23] IBM, ICU4J 3.4, International Component for Unicode for Java, Version 3.6. http://icu.sourceforge.net

[24] IBM, ICU User Guide, Transformation Rule Tutorial, script transliterator, http://www.icu-project.org/userguide/TransformRule.html

[25] Retrieved: http://wafakadard.com/

[26] Stories in Roman-Urdu, http://www.asian-women-magazine.com/fun/urdu-storeis.html

[27] MuziqPakistan, A Pakistani Forum, Online Discussion Community, http://www.muziqpakistan.com/board/index.php?showtopic=30380&pid=668827&mode=threaded&start=