World Applied Sciences Journal 22 (1): 113-120, 2013
ISSN 1818-4952

DOI: 10.5829/idosi.wasj.2013.22.01.142

Urdu Word Prediction System for Mobile Phones

Sana Shahzadi, Beenish Fatima, Kamran Malik and Syed Mansoor Sarwar

Punjab University College of Information Technology (PUCIT),
University of the Punjab, Lahore, Pakistan

Abstract: There are different issues with typing Short Messaging Service (SMS) text in mobiles. Different techniques have been proposed previously that help type faster. We present a new technique for this purpose that uses 'bigram' model developed for Urdu language word prediction. Data structures used are optimized and customized for mobiles phones, considering that they have small storage and less processing speed.

Key words: Word prediction · Dictionary based · Bigram model and mobile typing

INTRODUCTION

Urdu is the national language of Pakistan and one of the state languages of India. It is spoken by around 200 million people across the globe [wiki.answers.com]. Computational aspects of Urdu language started in early 1980s. Natural language processing (NLP) techniques are applied on Natural Language (NL) to obtain computable linguistic artifacts.

Short Message Service (SMS) is a text messaging service component of phone, web, or mobile communication systems that allows exchange of up to 160-character messages, including broadcast messages, between these technologies. Most of the SMS messages are mobile-to-mobile. According to the latest United Nations telecom agency (International Telecommunication Union-ITU) report, there are over 6 billion mobile phone subscribers in the world today [www.bbc.co.uk].

Most of Pakistan's 190 million people can speak Urdu and over 118 million (i.e., over 62% of the population) has access to mobile phones and SMS. SMS is a 20-year old (Naughton) and the most widely used data application of the world with trillions of SMS messages exchanged every year. With such a large number of text messages on mobile phones with reduced keypads and because some keypads have other shortcoming that user cannot easily adjust with them, the ease, comfort and assistant in typing become major challenges. The focus of our work is assistance in typing. We give details of previous techniques that solve different typing problems.

This document presents a bigram word prediction system in Urdu language and discuss in detail the technique proposed. Technique has made for mobiles that have less memory and less processing speed.

Section II discusses the existing work, section III gives detail about Corpus collection, section IV gives a brief introduction to the bigram model, section V gives details of the data structures used by our proposed system, section VI presents the adaption and scaling frequencies, section VII presents complexity of the scaling and adaption operations, section VIII presents our application's interface, section IX discusses efficiency of our system and section X discusses conclusions and outlines future directions.

Existing Work: Everyone appreciates fast typing and saving time using techniques that help typing fast. A number of techniques have been proposed and implemented in the literature in order to help faster typing on mobile phones.

User interacts with some hardware to enter input, e.g., keypads, keyboards, on-screen keyboards and keypads, stylus or through voice recognition system. Typing is a hectic and time-taking task, but there may also have been some obstacles in the device a user interacts with to enter text. It is a helpful and time-saving idea to use optimized systems and techniques to type faster.

Corresponding Author: Kamran Malik, Punjab University College of Information Technology (PUCIT) University of the Punjab, Lahore, Pakistan.

Traditionally, a mobile phone comes with a 12-key keypad on which 26 English characters as well as 39 Urdu characters mapped to nine keys. Since there are more characters than the number of keys, so more than one character have to share one key. This is why ambiguity arises when pressing a key, i.e., which of the characters mapped to the key should appear on the screen. A simple idea is to have multiple taps on the key in order to have the desired character on screen. Multi tapping is a solution but it is hectic and time-consuming and the user may get tired of tapping a lot of times. The 12-key keypad is still in use and comes with many popular brand mobile phones, including Samsung, Nokia, Motorola and Sonyericsson. Now days, mobile phones come with full key keypads that look like keyboards, but they are small in size. They resolve the multi tapping issue present in the 12-key keypad mobile phones. The use of mobile phones with both types of keypads is equally popular. Mostly, full-key keypad has QWERTY arrangement of characters (Noyes) and comes with the touch as well as non-touch mobile phones. Such phones are costly and usually have battery life of 6-9 hours with talk time.

Early optimization techniques that helped resolve ambiguity while typing are both hardware as well as software based. The simplest and baseline technique on 12-key keypad and reduced keypad is multi-tapping. Table 1 below contains the arrangement of characters on keypad for English.

In regular multi-tapping technique, first find the key on which the required character is located and then press the key and tap it one less times the index of character on that key, with the first character having index 0 [1]. For example, in order to type ‘v’ one needs to press 8 key once and tap the same key twice. Similarly, in order to type ‘m’, we simply need to press 6 key with no tapping. Character arrangement of Urdu language on reduced keypads is given in Table 2 and multi-tapping can be used for typing the relevant character on a key as explained above for typing English letters.

Ambiguities in reduced keypads are: (1) multiple characters share same key (2) typing two consecutive characters of a word that appear on the same key. Disambiguate these two using multi-tapping by (1) more than one tapings to select required character on the key (2) pause (timeout kill) to get another character from the same key respectively. Multi-tapping has three tasks that take time:

Table 1: Character arrangementof English on reduced keypad

1	./()…	2	abc	3	def
4	ghi	5	jkl	6	mno
7	pqrs	8	tuv	9	wxyz
	*	0	space		#

Table 2: Chracter arrangement of Urdu on reduced keypads

1	Symbols	2	ٹ ت پ ب	3	ء ا أ
4	ض ص ش س	5	ڈ ڈ ز ر	6	خ چ چ
7	ے ی ء ھ و ن	8	م ل ق ف	9	غ ع ظ ط
		0	Space		

- Moving finger to the required key
- More than one tapings to get the desired character types/displayed
- Pause to repeat-tap the same key for another character mapped to the same key

Multi-tapping takes time due to these three operations and that is bothersome to the user while typing. Silfverberg, MacKenzie and Korhonen [2] estimates time in seconds for these operations and calculate almost exact time of typing words using different techniques to compare them.

Gong and Tarasewich [3] gives solution to optimize the arrangement of characters on the keys so that the finger movement and the pause (timeout kill) time comparatively decreases while typing text. Constrained and unconstrained keypads differ from each other such that on a constrained keypad, letters are always arranged on keys in an alphabetical order, e.g., ‘a’ on the same key will always comes before ‘b’ or on the previous key on last position. Unconstrained keypads are not restricted to the alphabetical order. Constrained and unconstrained both keypads can be optimized to place characters that result in less pause time and less finger movement in order to type different characters from the keypad. The important thing to do is that frequent consecutive characters should not be on same key. Pause time can be reduced by examining text for consecutive letter frequencies for each pair of letters and mapping frequent consecutive letters on nearest keys but not on the same key. Thus, a user needs less finger movement to get to the next letter, which will be most of the time a frequent letter after the typed letter. Gong and Tarasewich [3] optimizes keypad for such pauses and finger movement time and gives three keypad designs in which one is unconstrained keypad design. All three keypads give good results. How and Kan [4], based on a study, estimates time in seconds that is used while typing for different operations such as

move and press key, repeat and press key and to move cursor in the text etc. According to Gong and Tarasewich [3], specific users may easily adjust to the constrained optimized design. How and Kan [4] uses a genetic algorithm to have an optimized keypad design and uses heuristics to come up with a design that reduces time for a list of typing operations. Letter Ease is also the same technique to reassign characters to keys by examining text for letter frequencies [5]. In Letter Ease, characters with higher frequencies are on starting keys and characters with less frequency are on later keys. In other words, characters are ordered on keypad according to their frequencies of occurrence in text.

Two-key text entry system uses a technique that each character takes exactly 2 keystrokes to enter [6]. User must press the key first having required character in the character list on that key and then press key to specify the position of that character on key. For example, to get 'n', press 6 and 2 and to get 'j', press 5 and 0.

Twiddler is a one handed remote type device to enter text [7]. Single keystrokes to different keys are needed to enter single character and no key is repeated to enter the same letter. Keystrokes needed for a letter to enter can be detected simultaneously. Lyons [7] reports that average results for twiddler are 60 words per minute. In order to type each character, a specific combination of keys must be followed by the user. See [7] for these key combinations.

Fastap [8], in contrast to the reduced keypad, has more physical keys. For each letter there is a single physical key that's why for each letter user need to press only one key once. To type word, the number of required keystrokes is equal to the length of the word. 26 keys are arranged on the keypad and each key is assigned a letter in alphabetical order. There are also keys for each digit and no two characters or digits share the key.

QWERTY is an international standard of arrangement of letters on keys. QWERTY keypad and keyboards are also similar to Fastap in that each key has only one character mapped to it and no character shares its key with any other character. The only difference is that arrangement of characters on keys is made in such an optimal way that after adequate user training it gives best typing results. There are techniques that adopt QWERTY such as half QWERTY and reduced QWERTY for special purposes.

There are some other techniques that are software based to optimize typing speed. These techniques are both dictionary based and non-dictionary based. Non-dictionary based techniques do not need to establish a dictionary of words, so these techniques did not deal with non-dictionary words while typing. Dictionary based techniques do not give prediction for non-dictionary words until the words are manually added to the dictionary by the user.

Dictionary based techniques use some type of corpus to build dictionary. T9 by Tegic Communications (www.tegic.com) and eZiText by Zi Corp (www.zicorp.com) are two commercial examples of such predictive text software that work by maintaining dictionary of words. T9 is used in most mobile phones for text entry. In T9, a user has to press corresponding key once for each letter with no tap and after typing, if the desired word does not appear then press a special key, usually *, to switch between different alternate words correspond to the same key character sequence [9]. This is letter-by-letter word prediction technique. The eZiText is a combination of T9 (letter-by-Letter) and word completion techniques. Word Completion feature suggests the user the most probable postfix character sequence automatically after each typed character sequence while typing. The suggestion of most probable postfix character sequence appears automatically after typed character sequence that can be selected by user using special key. Special key, usually *, is used to switch between postfix suggestions to get the required word on screen [9].

Non dictionary techniques, such as letter wise or prefix based technique, maintains a tree type structure for prefixes and frequencies related to each letter [10]. A finite automaton also may be used instead of tree type structure and each state represents a letter and transitions with the frequency to get any other letter after the given prefix letter sequence. For example, the most probable character to occur after prefix 'fo' is 'r' over the probability of occurrence of the characters p, q and s after 'fo'. In Prefix based technique, there are none of non-Dictionary words because data structure contains transitions for all characters with theirs probability of occurrence from each given prefix towards state of character.

Different methods are used to evaluate the performance of text entry techniques. Keystrokes per character and movement model Fitts' law [11] used to measure and compare the speed of different techniques. Fitts' law gives a more accurate measurement of speed in terms of time for each operation such as pause time in multi-tapping and finger movement time etc (see [2] for more detail). Silverberg, MacKenzie and Korhonen [2] compare three techniques using the movement model and

give results in terms of total words per minute. Number of keystrokes per character is not as accurate a measure as the movement model because it ignores the pause time and other small operations that take time. However, number of keystrokes per character is still used to measure the performance of text entry techniques.

Corpus Collection: We used training based technique where the system needs to train according to the given training data to learn to predict in the future. This data is called Corpus in NLP jargon. Corpus must be for the type of software we are going to develop, e.g., English dictionary needs English corpus, weather forecast needs real figures of previous weather and the prediction of stock exchange prices requires the related historical data of stock exchange. Mobile SMS text entry system needs data, which is normally used by the people in their messages. Our software had to be given as input common SMS data, which is similar to chat data. So, we designed our corpus using data from SMS sources and chat logs and then translated it to Urdu.

It was observed that mostly people talk in Urdu and English mix words, e.g., 'اوہےہ time ایک' i.e., English mixed with Roman Urdu. We, therefore, also added English words in our corpus in order to make our software more flexible so it could give predictions in Urdu as well as English.

An important statistics about our corpus is that it has about 2000 unique words. Normally, users use around 200-500 unique words from a dictionary to compose a text documents. We trained our software for approximately 250 commonly used words in sentences, including 'تھا', 'ہوا', 'تھی', 'کو', 'کیوں', 'کیا', from the system dictionary.

Bigram Model: System trained using bigram model that helps calculate probability for all bigrams in the corpus. The bigrams in the corpus means every two consecutive meaning full words in corpus. All consecutive bigrams occur in the corpus could write with together as they make sense with each other as in the corpus.

Bigram formula to calculate frequencies of all bigrams in the corpus is given below that helps to compare bigrams that the most probable bigram in the corpus will be the most suitable consecutive words that could write together. It does matter that which word in the bigram comes first (order of bigram words matters).

$$P(W_2|W_1)=\frac{count(W_2\ with\ W_1)}{count(W_1)} \quad (1)$$

In Equation 1, the probability of occurrence of W_1 will be calculated with the ratio between count of bigrams in which W_1 occurred before W_2 in corpus and the count of W_1 as a whole A system is trained against a given corpus for working using this formula by keeping counts. Our proposed system was trained using the Bigram model and worked according to the text for which it was trained.

Data Structures: Two data structures are used to maintain the dictionary. Word completion is also implemented along with word prediction using bigram model to cop up the efficiency of this technique along with the word prediction technique. The following is the list of the data structures maintained and used for different purposes.

We need to maintain a list of all unique words with associated index values in the Urdu corpus. Hashmap (also known as Hash Table) is not suitable because we did not need to store key-value pairs but a list of words that could be accessed with their associated index values. The best data structure for such storage requirements is a vector or an arraylist. We used a vector for this purpose since vector has function that could returns index of given word and vice versa. The example corpus is shown in Table 3 with corresponding vector in the Table 4 below.

Word prediction with bigram model calculates the number of consecutive occurrences of W_1 and W_2 in the corpus and stores occurrence count relative to their indexes in the arraylist of unique words maintained earlier. Although we could have used a two dimensional array because it gives access to the required element (i.e., word) in constant time, but it has the issue of wasted storage (sparse matrix) in our scenario because only limited number of words make sense to arise immediately after any given word from all possible unique words in dictionary. We must maintain a two dimensional array of size of N*N= N^2, where N is the number of unique words in dictionary. This is the wastage of space so, most of the two dimensional indexes will be unused and wasted. Since we are considering implementing the dictionary for mobile phones, that have small storage spaces, therefore, the use of a two-dimensional array would not have been efficient, although it would have given constant time accessibility to the occurrences of W_1 and W_2. We used a Hashmap with indexes of words as keys from the vector where each index referred to a unique word in the vector. The Hashmap stores information of the occurrence with pattern that is index of W_1as key will give a value of list. The list maintained a pattern that is 'total occurrence of

Table 3: Example Corpus

ں ی م ں ی ھب وت ں وھک ی د . ں ی م ں ی ھب ؤاج ی گ .

Table 4: Example Vector of unique words from corpus

ں ی م
ی ھب
وت
د ی ھک و ں
ج ؤ ں
گ ی

Table 5: HashMap of bigram frequencies for two words

Key: W_1	Value: Frequency of occurrence of W_1 with W_k
6	20; 5,7;2,4; 22,4; 6,3; 19,1;
7	40;11,10;23,9;33,8;41,6;90,6;

W_1; index of W_k, number of occurrences of W_k with W_1;' where W_k repeats for k (where $1 \leq k \leq N$) words that occur immediately after W_1 as shown in Table 5.

The given Hashmap states the format to store bigram frequencies of only two words that are located at index 6 and 7 in the vector. The original Hashmap will maintain the bigram frequencies for all the unique words that are stored in vector at each index. Length of Hashmap and vector will be equal because Hashmap has to keep track all the bigrams of all unique words from vector. In Table 5, 6 → 20; 5,7; 2,4; 22,4; 6,3; 19,1; indicates that the word at index 6 in vector occurs 20 times in the whole corpus such that 7 times with word at index 5, 4 times with the word at index 2, 4 times with word at index 22, 3 times with the word at index 6 and one time with the word at index 19. The use of Hashmap instead of 2 dimensional arrays saves memory by just increasing from constant to linear time complexity. The list of bigram frequencies is stored in descending order so that when we need to show prediction list just take first 4 to 5 words from that sorted list without searching the whole list to show most probable words in the popup menu.

For word completion we maintain a Hashmap with key value pairs. There are 39 possible characters in Urdu language and each word starts with character from one of these 39 characters. We maintained Hashmap such that key is the first character of word and resultant value will be the list of all words that starts with that character stored with their frequencies of occurrence in descending order as in the corpus. So the maximum length of Hashmap is 39. Since 39 length is constant in terms of space complexity. Hashmap maintained with all characters of Urdu as keys and stored information against them is the list of index of words from vector (in Table 4) starting with that specific character and the number of occurrences of that word in the corpus. Format to store word completion frequencies in dictionary is such that key is the character C_1 in Urdu character set and value against key is the list of indexes with associated frequencies from the corpus. Format of list against the given key is 'the number of occurrences of words that starts with character C1; index of Wk word that starts with character C1, number of occurrences of Wk;' where Wk repeats for all values of k where $1 \leq k \leq N$.

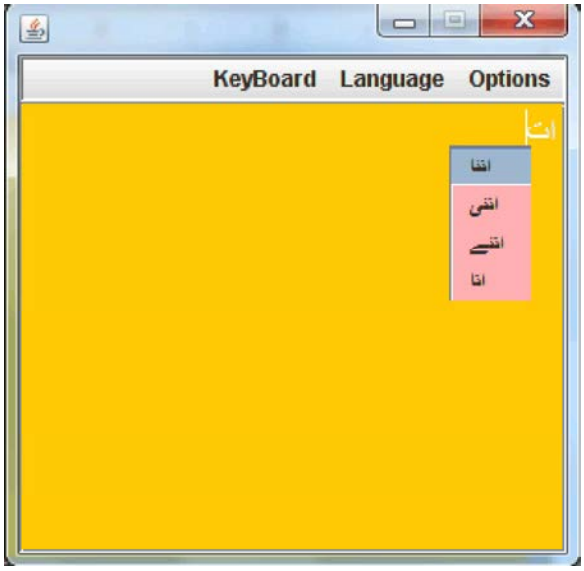
Table 6: Hashmap of word completion frequencies for two characters

Key: C_1	Value: Frequency of occurrence of W_K that starts with character C_1
ب	8;8,4;7,3;9,1;
م	7;44,3;19,3;15,1;

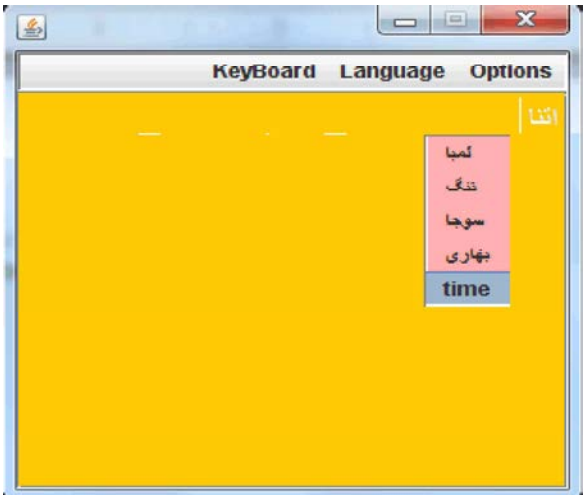
In Table 6, key of Hashmap represents character of Urdu character set and associated list is the indexes of all words in vector that starts with key character with corresponding frequencies in corpus. The character from Urdu character set 'ب' is a key and list against this key is '8; 8, 4; 7, 3; 9, 1;' where 20 indicates the total words including duplicates that starts with key character such that word at index 8 in vector that starts with character 'ب' has 4 occurrences in the corpus, word at index 7 in the vector that starts with character 'ب' has 3 occurrences in corpus and word at index 9 in the vector that starts with character 'ب' has 2 occurrence in corpus.

Adaption: The word prediction technique is for the mobile SMS system that will work in mobile for a single user at a time. Thus, the technique must adapt to the user using the mobile phone. The dictionary would be populated with words as a result of training but users have their own vocabulary of words for daily use that must add to dictionary. Thus, an option given to the user to add user vocabulary (i.e., non-dictionary words) should be turned on automatically otherwise the system would prompt to ask whether the typed non-dictionary words should be added to the dictionary or not. Similarly, there is option to turn off dictionary predictions and switch to traditional multi tapping.

Adapt and change according to user's preference means that words used by user will be preferred by the system. Words will sorted according to theirs frequencies and first five will be in prediction list and others all will be called as non prediction list words. If user type words that is a non prediction list word, system change position of that word to 1/3 higher in non-dictionary words and assign frequency of higher position word to that word and all other words will be shifted one position lower. After typing three times by the user that word will be



Word Completion



Word Prediction using bigram

added to the prediction list words that is adaption of system according to user's preference. Once the word included in prediction list words the change of probabilities will be normal and frequency of word increased by one of w_2 with each usage after W_1.

Linear Operations: Adaption and scaling take linear time when the data structures have any change or updating operations.

- As new word appear in text, the word added to vector in constant time and key value pair in bigram Hashmap in constant time and to add in word completion dictionary takes linear time to insert in sorted frequency list.
- Updating existing frequencies need to move in sorted list of frequencies so that resultant list also is sorted. Insert into sorted list takes linear time including movement of existing elements.
- Scaling takes linear time for a single list stored in Hashmap against a key, linear time as total words (N is total words) that could come after W_1 will be k where $1 \leq k \leq N$.
- Adaption takes linear time because it is an operation to adapt according to user-specified, moving a word to 1/3 upper position in the list of words will eventually add that specific word to prediction list words so that after 2 to three times manual typing word preferred by system as user specify.

Interface: English keyboard characters are mapped to Urdu characters. Phonetic based mapping is performed on Urdu-to-English characters for ease of learning, for example, the letter 't' has been mapped to the Urdu character 'ب'. The keyboard mapping has been attached to the software for usability point of view. Predicted values are shown when user type. On pressing <Space> key predicted values are show. When user type first letter prediction, the algorithm start working. If user does not select result for option then after prediction will be shown after next typed letter. The following screenshots show these two interfaces of our software.

RESULTS

No. of Keystrokes presses to type a message are used. Test data of 12000 words is used. For the given 12000 words the user had to type only 3772 words and remaining 8228 words were typed by our system's prediction technique. This figure is maintained with the first usage after training without giving time to system to adapt according to the user.

CONCLUSION

Word prediction using the bigram model and merging that technique with word completion gives better results over the previously proposed techniques. Most common words used in the language such as stop words will always be there in dictionary and obviously will have higher frequencies of occurrence to select from list because their ratio will always more than non-stop words. Bigram word prediction will provide many words without a single stroke by just selecting from the list, which is the best aspect of our proposed technique. Word completion

technique facilitates users when word prediction technique's prediction does not contain word or have low priority to select from automated generated prediction list.

Our technique is purely designed for such devices having less memory and processing power. It is, therefore, optimized and customized to occupy less memory and computation such as searching but sorting, scaling and adaption has linear complexity with respect to the total number of unique words in the input list.

We intend to address the following issues in our future work:

- Parts of speech tagging and Noun and verb phrase chunking can be used to make prediction list more precise and accurate that releases user confusion [12-15].
- With the continues and frequent use of dictionary may result in high frequencies for some words may scale to lower level without affecting probabilities.
- Measure performance results of a word prediction technique after combining our technique with T9 letter-by-letter word prediction instead of word completion.
- Conduct a study on using a mixture of already proposed techniques for best fit and more optimized results.
- Use semantic knowledge, word structure and meanings can also be used for more accuracy in prediction.
- Add transliteration that transliterate roman English to Urdu easily and automatic correct spelling errors [16].
- By using behavior of word Kaa [16] we can improve its performance.

REFERENCES

1. Yang, D., *et al.,* 1999. Reduced Keypad Entry Apparatus and Method. Google Patents.
2. Silfverberg, M., I.S. MacKenzie and P. Korhonen, 2000. Predicting Text Entry Speed on Mobile Phones. Proceedings of the SIGCHI conference on Human factors in computing systems: ACM, pp: 9-16.
3. Gong, J. and P. Tarasewich, 2005. Testing Predictive Text Entry Methods with Constrained Keypad Designs. Citeseer.
4. How, Y. and M.Y. Kan, 2005. Optimizing Predictive Text Entry for Short Message Service on Mobile Phones. Proceedings of HCII, pp: 5.
5. Ryu, H. and K. Cruz, 2005. Letterease: Improving Text Entry on a Handheld Device Via Letter Reassignment. Proceedings of the 17th Australia conference on Computer-Human Interaction: Citizens Online: Considerations for Today and the Future: Computer-Human Interaction Special Interest Group (CHISIG) of Australia, pp: 1-10.
6. Kandogan, E. and S. Zhai, 2004. Two-Key Input Per Character Text Entry Apparatus and Method. Google Patents.
7. Lyons, K., *et al.*, 2004. Twiddler Typing: One-Handed Chording Text Entry for Mobile Phones. Proceedings of the SIGCHI conference on Human factors in computing systems: ACM, pp: 671-78.
8. Cockburn, A. and A. Siresena, 2003. Evaluating Mobile Text Entry with the Fastap Keypad. University of Canterbury. Computer Science and Software Engineering.
9. Forcada, M.L., 2001. Corpus-Based Stochastic Finite-State Predictive Text Entry for Reduced Keyboards: Application to Catalan., 27: 65-70.
10. MacKenzie, I.S., *et al.*, 2001. Letterwise: Prefix-Based Disambiguation for Mobile Text Input. Proceedings of the 14th annual ACM symposium on User interface software and technology: ACM, pp: 111-20.
11. Zhai, Shumin, Alison Sue and Johnny Accot, 2002. Movement Model, Hits Distribution and Learning in Virtual Keyboarding. Proceedings of the SIGCHI conference on Human factors in computing systems: Changing Our World, Changing Ourselves: ACM, pp: 17-24.
12. Fareena Naz, Waqas Anwar, Usama Ijaz Bajwa and Ehsan Ullah Munir 2012. Urdu Part of Speech Tagging Using Transformation Based Error Driven Learning. World Applied Sciences Journal, 16(3): 437-448.
13. Shahid Siddiq, Sarmad Hussain, Aasim Ali, M. Kamran Malikand and Wajid Ali, 2010. Urdu Noun phrase chunking: Hybrid approach, In proceedings of IEEE 2010 International Conference on Asian Language Processing, Harbin China, pp: 28-30.
14. Aasim, Ali, Sarmad Hussain, M. Kamran Malikand and Shahid Siddiq, 2010. Study of Noun Phrase in Urdu, in proceedings of Conference on Language and Technology 2010, Society of Natural Language Processing Pakistan, Islamabad Pakistan, pp: 22-24.
15. Wajid Ali, M. Kamran Malik, Sarmad Hussain,

Shahid Saddiq and Aasim Ali, 2010. Urdu Noun phrase chunking: HMM based approach, In the proceedings of IEEE 2010 International Conference on Educational and Information Technology (ICEIT 2010), Chongqing China, 17-19 September, ISBN: 978-1-4244-8034-0, 2: 494-497.

16. Kamran Malik, M., Aasim Ali and Shahid Siddiq, 2010. Behavior Of Word 'Kaa' In Urdu Language, In proceedings of IEEE 2010 International Conference on Asian Language Processing, Harbin China, pp: 28-30.