A Hybrid Approach for Urdu Spell Checking

MS Thesis

Submitted in Partial Fulfillment
of the Requirements for the
Degree of

Master of Science (Computer Science)

at the

National University of Computer & Emerging Sciences

by

Tahira Naseem

November 2004

Approved:

_____

Dr. Fakhar Lodhi
Head
Department of Computer Science

_____20 ____

Approved by Committee Members:

Advisor

_____

Dr. Sarmad Hussain

Associate Professor

FAST - National University

Other Members:

_____

Mr. Shafiq-ur-Rahman

Associate Professor

FAST - National University

Vita

Ms. Tahira Naseem was born in Mandi Baha-ud Din, Pakistan on November 17, 1980. She received a Bachelor of Science in Computer Science from Lahore College for Women University, Lahore in 2002. The research in this dissertation was carried out from 2002 to 2004.

Her interests include studying books of every kind except course books.

Acknowledgements

First of all I want to express my deepest gratitude to Allah Mian who gave me the strength and ability to accomplish this research work.

I am very thankful to Dr. Sarmad for his consistent support and valuable suggestions. In addition he also persuaded me to complete this work in due time, for which I am really grateful. I would also like to thank my advisor Mr. Shafiq-ur-Rahman for his guidance.

I also want to acknowledge Rubina for her assistance in very tedious task of finding spelling mistakes in Newspapers. Special thanks to Huda Sarfraz, who proofread my proposal defense document with very keen attention.

Finally I want to thank my parents whose prayers are always a great source of strength for me.

Table of contents

1 Introduction

Problem of automatic spell checking is not new in the areas of Information Retrieval and Language processing. The research started as early as 1960s [Damerau 1964]. Many different techniques for detection and correction of spelling errors are proposed during last 40 years. Some of these techniques exploit general spelling errors trends while others use the phonetics of misspelled word to find likely correct words. In recent years statistical based techniques, which are not explicitly based on error trends but through training, adapt to error patterns, have got more popularity [Kukich 1992].

Quite a few of these techniques are being used with text editors and other text handling applications and are showing reasonably good performance. Nevertheless the problem of spell checking is still considered open for further research and improvements. There are mainly two reasons for considering this problem still unsolved; one has to do with the performance of existing techniques and the other with their scope.

The first reason is that as the research in the area of Natural Language Processing advanced over the years, the need of automated spell checking is being felt for many tasks other simple proof reading of computer generated text. Many NLP applications like Machines translation Systems, Text to Speech Systems and Optical Character Recognizers require automated spell checking of text. The demands that are implied by these applications are much more challenging than the ones implied by human users of spellcheckers. The major difference is that, for a human user it is adequate if the errors are detected and for every error a small number of suggestion are proposed from which user can select the required one; on the other hand in automated spelling correction it becomes spellchecker's responsibility to decide on what is required, spell checker should be able to find one best correction for an error with ideally 100% accuracy. This level of accuracy has not yet been achieved.

The second reason for considering the spell-checking problem unsolved is that most of the techniques proposed so far are based on English or some other Latin script based language. Since every language has its own writing system, the techniques that perform well for one language may not perform that well for some other language, they may even totally fail, for example if English spellcheckers are tested on text of Thai language they will fail on very first step of recognizing word boundaries because in Thai, unlike English, word boundaries are not marked with spaces; The writing system of a language also governs the types and trends of spelling errors of that language. Therefore existing techniques, which are designed mainly focusing English language, are limited in their scope.

From this discussion it can also be conjectured that in order to propose a new spell checking technique or fit an existing one on a language having writing system significantly different from English, one has to clearly identify the language specific issues and deeply investigate general spelling error trends of the language, only then a reasonably effective spell checking approach can be proposed.

Urdu is also among the languages whose writing system is different from that of English and therefore existing techniques cannot be applied for Urdu spell checking without modifications. This document presents the details of a study performed on Urdu language to identify the problem areas of Urdu spell checking and to test the effectiveness of existing spell checking techniques on Urdu. A hybrid approach for Urdu spelling error correction is also proposed which uses a combination of more than one existing techniques. These techniques are modified to cater Urdu specific problems and to make use of error patterns in Urdu.

The document comprises of the five sections. First section provides an overview of the work done so far in the area of spell checking, next section is on Urdu Spell Checking in which Urdu related spell checking issues are discussed, next is the Problem Statement section in which the statement of the problem for this research work is given, next comes the Methodology section in which the methodology of the research work and the results are discussed and finally in the Conclusion section where results are summed up and some points for future research are suggested.

2 Background

Before going into details of different spell checking approaches it will be a good idea to first have knowledge of what is the spell-checking problem? What are its major aspects? And what are the different types of spell correction.

The functionality of a spell checker is so common that defining it formally is unnecessary, but a distinction must be made between two core functionalities provided by a spellchecker i.e. spelling error detection and spelling error correction. 'Error Detection' is to verify the validity of a word in the language while 'Error Correction' is to suggest corrections for the misspelled word.

Spelling error correction can be of two types, 1) interactive, and 2) automatic. In interactive the spellchecker can suggest more than one correction for each error and the user has to select one for replacement; in automatic correction, the spellchecker has to decide on the one best correction and the error is automatically replaced with it. Automatic error correction is the requirement for those speech processing and Natural Language Processing (NLP) related systems where human intervention is not possible [Kukich 1992].

The spell checking process can generally be divided into three steps, 1) detecting errors, 2) finding correction and 3) ranking correction. Detection and correction are already defined; Ranking is the ordering of suggested corrections in decreasing order of their likelihood for being actual intended word. Different techniques can be used on each step or the same technique can be applied to all three merging them into one step.

In the following sections, general types and trends of spelling errors and various techniques for detection and correction of spelling errors are discussed.

2.1 Types of Spelling Errors

Until recently, most of the spelling correction techniques were designed on the basis of spelling errors trends (also called error patterns); therefore many studies were performed to analyze the types and the trends of spelling errors. Most famous among them are the studies performed by Damerau (1964) and Peterson (1986). The majority of the early research done in the area of spell checking was based on these studies.

According to theses studies Spelling errors are generally divided into two types, typographic errors and cognitive errors.

2.1.1 Typographic errors

Typographic errors occur when writer knows the correct spelling of the word but mistypes the word by mistake. These errors are mostly related to the keyboard and therefore do not follow linguistic criteria.

A study by Damerau (1964) shows that 80% of the typographic errors fall into one of the following four categories

1. Single letter insertion; e.g. typing acress for cress
2. Single letter deletion, e.g. typing acress for actress
3. Single letter substitution, e.g. typing acress for across
4. Transposition of two adjacent letters, e.g. typing acress for caress

This statement was confirmed later by a number of researchers including Peterson (1986), Mitton (1987).

The results of a study by [Peterson 1986] are shown in Table 1. The data sources were Webster's Dictionary and Government Printing Office (GPO) documents retyped by college students.

	GPO	Web7
Transposition	4 (2.6%)	47 (13.1%)
Insertion	29 (18.7%)	73 (20.3)
Deletion	49 (31.6%)	124 (34.4%)
Substitution	62 (40.0%)	97 (26.9%)
Total	144 (92.9%)	341 (94.7%)

Table 1. Comparison of four basic types of errors.

The rows in Table 1 correspond to four basic types of errors; the columns correspond to the two sources of data. For each data source, the number and percentage of each type of errors is given. The last row contains total number and percentage of single errors.

Typographic errors are mainly caused due to *keyboard adjacencies*. The most common of these typographic errors is the substitution error (as shown in 4th row of Table 1). Substitution error is mainly caused by replacement of a letter by some other letter whose key on the keyboard is adjacent to the originally intended letter's key. In a study referred to by Kukich (1992), 58% of the errors involved adjacent typewriter keys.

Insertion errors can occur due to the double pressing a key or by accidentally hitting two adjacent keys while trying to hit one of them. [Damerau 1964]

Omission errors usually occur when the eyes move faster than the hand. [Damerau 1964]

According to Peterson (1964) the next most common errors are two extra letters, two missing letters and transposition of two letters around a third one.

2.1.2 Cognitive errors

Cognitive errors (also called orthographic errors) occur when writer does not know or has forgotten the correct spelling of a word

It is assumed that in the case of cognitive errors, the pronunciation of misspelled word is the same or similar to the pronunciation of intended correct word. (e.g., receive → recieve abyss → abiss)

In a study referred to by [Kukich 1992], Dutch researchers let 10 subjects transcribe the 123 recordings of Dutch surnames, 38% of these transcriptions were incorrect despite being phonetically plausible. In another study referred to by [Kukich 1992], done on spelling errors trends in students of different grades, considering only those mistakes whose frequency was greater than 5, it was found that 64.69% were phonetically correct and another 13.97% were almost phonetically correct. It was postulated that errors with lower frequency have a tendency to be less phonetic.

From the above discussion on types and patterns of spelling mistakes it is evident that there is a reasonable likelihood of both types of mistakes and a technique can successfully correct spelling errors only if it take care of both types of mistakes.

2.2 Implementation aspect of a spell checker

A spell checker is comprised of mainly two modules, a Lexicon (or Dictionary) and bunch of algorithms or techniques that use this lexicon for spell checking. These techniques generally provide three types of functionalities:

1. Lexicon lookup i.e. error detection
2. Finding approximate matches in lexicon i.e. Error correction
3. Ranking of corrections.

Some spellchecker use same technique to provide all three functionalities in one step, while others use different technique for each functionality. Lexicon lookup techniques are generally dependent on the structure of the lexicon, while on the other hand the design of storage structure is some times governed by the choice made for error correction technique. Therefore techniques for above mentioned functionalities are highly dependent on each other and on the structure of the Lexicon. In next two sections error correction techniques and lexicon storage structures are discussed. Error detection techniques are discussed in both sections since every storage structure readily suggests a corresponding lookup mechanism and every correction technique can essentially be used for detection as well. Ranking techniques are not discussed separately rather they are explained wherever they naturally appear in the flow of discussion.

2.2.1 Spelling Error Correction

Spelling error correction can be of two types: Isolated-Word Error Correction, Context Based Error Correction.

In Isolated-Word Error Correction, a misspelled word is analyzed in isolation without giving any consideration to its context; therefore the corrections are based only on the misspelled word itself. In **context-based correction**, on the other hand, the context of error is also taken into consideration for suggesting and ranking corrections. This second approach is particularly useful for correcting **real-word** errors. Real-Word errors are those spelling errors, which result in a valid words of Language that are not the actual intended words, for example writing “form” for “from” “مجرم” for “محرم” or “دغا” for “دعا”.

ایک نقطے نے ہمیں محرم سے مجرم بنا دیا

ہم دعا لکھتے رہے وہ دغا پڑھتے رہے (غالب)

Such errors can never be caught without using contextual information.

Contextual information can be used for ranking the suggested corrections, especially when more than one suggestions otherwise seem equally likely for being the actual correction.

Early work in the area of spell checking was more focused on isolated-word error correction, but with the passage of time, the number of such applications increased where auto-correction was a requirement, for example in applications like Text to Speech Synthesis systems, Machines Translation systems or other NLP related systems.

In such applications the spellchecker should be capable of catching real word errors. Moreover it should also be capable of deciding one best correction, and this can be achieved only if the context information is also used for correction.

Techniques for the corrections of these two types, i.e. isolated word error correction and context sensitive error correction, are discussed separately in the following two sections.

2.2.1.1 Isolated-Word Error Correction

In last 40 years, many different techniques have been invented for isolated word error correction. A good many of these techniques have been successfully implemented and are being used in different applications, but none of them has yet been succeeded in achieving near 100% accuracy without human involvement.

Most of the earlier word was rule based. Error patterns were used for finding corrections. This includes the work done by Damerau (1964), Angell et. al. (1983) and Zobel (1994). In the recent works probabilistic models are used for error correction which instead of subjectively making use of error patterns, use the erroneous data to automatically train the model according to the error patterns present in training data [Curch et. al. 1991], [Brill and Moore 2000], [Toutanova and Moore 2002].

Generally Isolated Word Error Correction techniques can be divided into following subcategories:

1. Edit distance techniques
2. Phonetics based techniques
3. Similarity Key techniques

4. N-Grams Based Techniques
5. Probabilistic Techniques

This division is not completely disjoint or distinct. The techniques in these groups may have overlapping features.

2.2.1.1.1 EDIT DISTANCE TECHNIQUES

The term 'Edit Distance' was originally defined by [Wagner 1974] as the minimum number of editing operations required to convert one string to another string. A similar concept was implemented about ten years earlier by Damerau (1964). [Kukich 1992]

Damerau's Single Errors Technique

[Damerau 1964] showed that 80% of spelling errors belong to one of the four classes of single errors.
These classes are:

Single letter insertion
Single letter deletion
Single letter substitution
Transposition of two adjacent letters

On the basis of this observation he proposed Single Error spelling correction technique. In this technique for an erroneous word all dictionary words are checked to see if the error could be formed from them applying any of the above-mentioned four operations. The words that passed this test are considered as possible correction. The technique showed 84% accuracy when tested on a test set of 964 errors.

Gorin [1971] applied this misspelling mechanism in reverse to find possible corrections for misspelled words. He generated all those words for a misspelling from which the misspelled word could be derived by applying any of the four error operations mentioned above.

This means that if the length of a misspelled word is n and the number of alphabets in the language is m then there will be n possible deletions, n-1 transpositions, (m-1)n substitutions and m(n+1) insertions. Thus a total of (2m+1)n+m-1 words will be generated. For English this number becomes 53n+25 and for Urdu, if we take total number of alphabets to be 43, it becomes 87n+42.

Once the words are generated they are tested against dictionary for being correct words in the language. In order to increase the efficiency of this correctness test, an optimization was proposed and implemented by the designers of a Swedish spellchecker STAVA [Kann, Viggo et. al. 1998]. Prior to dictionary look up, they tested the words using N-grams tables. These tables contain all those character sequences of length N that are allowed in the language. If a word contains a sequence that is not allowed in the language, the word is rejected without consulting dictionary.

Single Error technique is the only technique that works on the misspelled word to find possible correction; all other techniques process all dictionary words to find the closest match.

In this technique the proposed words cannot be ranked as most or least likely corrections. The suggestions are all at the same level.

Levenshtein Distance

The Levenshtein distance technique, like Damerau's single spelling error technique, measures the distance between two character strings in terms of insertion, deletion, substitution and transposition, but it is comparatively more generic because it allows multiple errors. [Erikson 1997]

To measure the Levenshtein distance between a misspelled word and any correct dictionary word all possible conversions between the two string applying single editing operations (insertion, deletion, substitution and transposition) are considered, this is done using a dynamic programming algorithm presented by Levenshtien (see Appendix A for the details of algorithm), the minimum number of editing operations required for conversion is found out. This number is the Levenshtein distance between the two strings. The greater the Levenshtein distance, the more different the strings are. For example Levenshtein distance between بادر and بیدار is 2, because one substitution and one insertion are required to convert بادر to بیدار،

بادر	→	بیدر	Substitution
بیدر	→	بیدار	Insertion

The dictionary word that is at the shortest distance from the misspelling is suggested as the most probable correct word. The words beyond a pre-specified threshold edit-distance are ignored.

In a modified edit distance technique, contribution of transposition operation, to the distance, is double the contribution of other operations taking it as a combination of insertion and deletion.

Weighted Edit Distance

In Damerau and Levenshtein edit distance techniques it is assumed that all the letters are equally probable for insertion, deletion or for substitution for some other letter, but in fact this is not the case. Due to keyboard configuration and phonetic similarities, inter-substitution of some pairs of letters is much more probable than others.

A study referred to by [Kukich 1992] shows that 58% of substitution errors are due to adjacent keyboard keys.

To make use of these facts, confusion matrices are constructed. These are n*n matrices, where n is the size of language alphabet. Each *ij* entry of the matrix is the probability of replacement of i^{th} letter by j^{th} letter. [Erikson 1997]

Similarly, confusion matrices are constructed for omission and insertion. These matrices contain the probabilities of insertion or omission of a letter on the basis of its left context.

Techniques of this type are discussed in detail under the heading of noisy channel model in Section 2.2.1.1.5.

Tapering is also a refinement on edit distance techniques. This technique assigns greater penalty to the errors in the beginning of the word i.e. two words are less similar if they differ by a letter in the beginning of the word and more similar if different letter is at the end of the word. [Zobel & Dart 1994] e.g. سوار is more similar to سوال than خوار

2.2.1.1.2 N-Gram similarity measure

An N-gram is a sequence of N adjacent letters in a word where N can be 1, 2, 3…. An N-gram is called a bigram or digram when N=2 and a trigram when N=3.

N-grams are used to measure similarity scores between two strings. The more N-grams they share the more similar they are. In a method given by Pfeifer (1995) a similarity coefficient δ can be calculated by dividing the number of common N-grams of the two strings by the total number of N-grams in two strings. Bigrams are most commonly used, along with leading and trailing blank. This addition of blanks result in assignment of extra importance to start and end of the word, as the starting and ending letter will individually form bigrams. [Holmes][Erikson 1997]

For بالand بالی

Bigrams in بالی :	ی-	لی	ال	با
ب-		5 bigrams		
Bigrams in بال :	ل-	ال	با	ب-
	4 bigrams			
Union :	ل-	ی-	لی	ال
با	ب-	6 bigrams		
Common :	ال	با	ب-	
	3 bigrams			

δ =3/6=.5.

N-gram techniques do not show good performance on short words. For example when using trigrams, the words of length three will share no trigram with themselves just after introduction of a single error. To overcome this drawback, N-grams of different lengths are used for words of different lengths. For very short words of length three or less, unigrams are used [Kukich 1992].

A study done by [Angell et al. 1983] shows that N-gram similarity measure works best for insertion and deletion errors, well for substitution errors, but very poor for transposition errors. [Kukich 1992]

2.2.1.1.3 Phonetics Based Techniques

These techniques work on the phonetics of the misspelled string. The target is to find such a word in dictionary that is phonetically closest to the misspelling.

Soundex

Soundex was the first phonetic string-matching algorithm developed by Odell and Russell (1918). It was originally developed to match names. The idea was to assign common codes to similar sounding names.

The length of the code is four and it is of the form *letter*, *digit*, *digit*, *digit*. First letter of the code is same as is the first letter of the word. For each subsequent consonant of the word, a digit is concatenated at the end of the code. All vowels and duplicate letters are ignored. The letters *h*, *w* and *y* are also ignored. If the code reaches the maximum length, extra characters are ignored. If length of code is less than 4, zeroes are concatenated at the end. Digits assigned to the different letters for English are shown in Table 2.

1	b, f, p, v
2	c, g, j, k, q, s, x, z
3	d, t
4	L
5	m, n
6	R

Table 2. Soundex Codes

Sample codes:

Codes for “Robert” & “Robin”

R→R o→_ b→1 e→_ r→6 t→3

Robert→ R163

R→R o→_ b→1 i→_ n→5

Robin→ R150 (an extra 0 is appended in the end to complete 4 digits)

Codes for “Smith” and “Smyth”

S→S m→5 i→_ t→3 h→_

Smith→S530

S→S m→5 y→_ t→3 h→_

Smyth→S530

This Soundex algorithm was actually designed for names; therefore its performance for spell-checking is not very good. A study by [Stanier 1990] shows that even for name searching, one fourth of the relevant words go undetected and only one third of the detected words are actually relevant. The large size of group 2 might be a reason for this poor performance.

Soundex was refined for use in spelling correction. The major change was the break down of group 2 into smaller groups. Refined Soundex is shown in Table 3. [Zobel and Dart 1994]

1	b, p
2	f, v
3	c, k, s
4	g, j
5	q, x, z
6	d, t
7	L
8	m, n
9	R

Table 3. Refined Soundex

The Soundex algorithm had many problems. First, it gives no importance to the sounds produced by letter groups, for example, sounds produced by *ch* and *sh*. Second, it retains first letter of the word. If an error occurs word-initially the Soundex will not be able to correct it. Different studies show that chances of an error at first position of word are very small. [Peter 1998]

To solve these problems many variants of Soundex were designed which used different length codes, multiple codes for same word and overlapping groups of alphabet. Those letters, which could produce more then one sound, for example c and g, were included in more than one groups. As a result multiple codes can be generated for same word. This increased the accuracy of Soundex. Code length also affects the performance. Codes with smaller length increase hit rate and codes with larger length result in greater accuracy.
Some of the Soundex variants are discussed later in this document.

Blair's Abbreviations

Blair invented an abbreviation-based technique in which all dictionary words and the misspellings are abbreviated. Each dictionary word is abbreviated to 4 letters. Weights

are assigned to all letters and to position of letters in the word on the basis of how desirable it is to delete some specific letter or a letter in some specific position. First letter position has a weight 1 the last position has weight 2 second position has weight 3 and so on. To abbreviate a word its Letters are arranged in decreasing order of their weights (letter weights multiplied by position weights), and then all letters except the last four are deleted. If the abbreviation results in a collision the length is increased to 5 and so on until all dictionary word are uniquely abbreviated. In this technique only those words are considered similar whose abbreviations match exactly. [Alberga 1967]

Damerau (1964) made a comparison of this technique with his own single error techniques and results showed that they performed almost equally well with human made (orthographical) errors but abbreviations technique performed very poorly for the data in which errors were introduced due to equipment malfunction. [Damerau 1964]

Phonix

Phonix is a Soundex variant. Like Soundex, it assigns codes (slightly different from Soundex, see Table 4) to letters but prior to code generation, string is standardized by applying some letter groups' substitutions. This process is also called N-gram substitution. About 160 letter group transformations are used. For example the sequence *tch* is mapped to *ch*, *ph* is mapped to *f*. In some implementations, different transformation rules are written for same letter group in different positions. [Holmes][Erikson 1997][Zobel and Dart 1994] [Pfeifer et. al. 1996]

1	b, p
2	c, g, j, k, q
3	d, t
4	L
5	m, n
6	R
7	f, v
8	s, x, z

Table 4. Phonix codes

Pfeifer et. al. (1996) tested Soundex and some Phonix variants in combination with Edit distance and Bigram similarity measure. They reported 80% precision and 80% recall. (Precision is the percentage of relevant words in the retrieved ones and recall is the percentage retrieved relevant words in all relevant words).

Editex

Editex is a combination of edit distance technique and letter grouping property of Soundex and Phonix. Editex forms overlapping groups of letters. To compute edit distance for Editex, 0 is added to edit distance if the corresponding letters of two strings are similar 1 is added if they belong to the same 'Editex-group' and 2 otherwise. This approach assigns smaller edit distance value to phonetics based substitution edits as compared to any arbitrary substitution. Letter groups of Editex are shown in Table 5. [Zobel and Dart 1994]

1	b, p
2	c, k, q
3	d, t
4	L, r
5	m, n
6	g, j
7	F, p, v
8	S, x, z
9	c, s, z

Table 5. Editex letter groups

2.2.1.1.4 Similarity Keys

Similarity key techniques, like Soundex and Phonix, assign codes to the words but the algorithms used for code assignment are different. These techniques are used in combination with edit distance techniques.

Skeleton Key

Skeleton Key of a word is formed by concatenating the first letter of a word with following consonants in order of their appearance in the word followed by the vowels of the words in the order of their appearance. Duplicate consonants are dropped. This technique like Soundex assumes that first letter errors are less likely to occur; therefore it cannot correct first letter errors. [Erikson 1997]

Omission Key

Statistics show that omission frequency of different English letters is different; some letters are omitted more frequently than others. Following are English consonants arranged in decreasing order of their omission frequencies:

RSTNLCHDPGMFBYWVZXQKJ [Pollock & Zamora 1984]

Omission Key uses these statistics. It is formed by concatenating the word's consonants in the reverse of above order and then vowels are concatenated in their original order. [Erikson 1997]] [Pfeifer et. al. 1996]

Plato Key

Plato Key is a numeric key that contains information about the word's length, letter contents, letter order and syllabic pronunciation in the same key. This technique is flexible because new features can easily be added to the key. [Erikson 1997] [Kukich 1992]

2.2.1.1.5 Spelling Error Correction using Noisy Channel Model

All the techniques that we have discussed so far rely on some sort of distance or similarity measure to find closest match. A drawback of using such measures is that certain important factors, which affect the error patterns, are ignored, Since spelling error trends depend on many different factors and these similarity measures make use of at most one (or sometimes none) of them, these measures become biased or insufficient. For example phonetics based techniques give no weight to keyboard adjacencies, according to the Soundex code 'find' can not be a candidate correction for 'gind' because they have different codes, although there is a significant chance that f is substituted by g due to keyboard adjacency.

There are similar problems with edit distance and N-Gram Techniques. For example in both edit distance and N-Gram techniques, لحاظ and لحاف are equally probable corrections for misspelling لحاض, which we can see is not true. There is a greater chance of confusing 'ض' for 'ظ' , due to sound similarity.

Considering these problems it can be seen that what is actually needed is a technique that can model the actual phenomenon of making spelling mistakes. One way of doing this is to use our knowledge of spelling error trends and of the factors affecting spelling errors

patterns and then devise a technique in which weights are subjectively assigned to these factors. One such effort was made by [Kashyap & Oommen]. They tested their technique on artificially generated errors in words of short length, and reported accuracy ranging between 30% and 92% depending on the word length.

The problem with this methodology is that we can never be sure whether our information about error patterns is exhaustive or not. Secondly even if we have complete knowledge of these factors, modeling them subjectively requires too much effort and even then the resulting system will neither be very fine, nor will be adaptive for different domains.

Another more generic and perhaps more appropriate way of doing this is to construct a probabilistic error model which can be trained for different languages and domains thus automatically setting its parameters. This is what is called 'Noisy Channel Model' of error correction. The idea behind this name comes from considering the phenomenon of making spelling mistakes as the process of sending text through a noisy communication channel, which introduces errors in the text. Our task is to find the most probable transmitted word (correct dictionary word) for a received erroneous string (misspelling). Now if the behavior of the noisy channel is properly modeled, a correct guess can be made of actual intended word by decoding the error pattern. [Jurafsky 2000] [Kukich 1992] [Brill & Moore 2000].

The model assigns a probability to each correct dictionary word for being a possible correction of the misspelling. The word with highest probability is considered the closest match (or the actual intended word).

This way of identifying the actual form from an observed or surface form is called Bayesian classification. It has been successfully applied in many speech and language processing applications in which some sort of identification or classification is to be made on the basis of incomplete or ambiguous information. [Jurafsky 2000]

Formally the problem can be stated as follows,

Let D be a dictionary and w_i be any word in D, for a misspelled string $s \notin D$ our task is to find such $w \in D$ for which P(w|s) is maximum. Hence,

$$w = \text{argmax}_{wi} P(w_i|s) \qquad (1)$$

Applying Bayes' rule we can rewrite this probability expression as

$$P(w|s)=(P(s|w)P(w)) / P(s) \quad (2)$$

Since we are maximizing P(w|s) over all dictionary words for our single observation string s, we can ignore P(s), as it will remain constant during single correction process. Hence to maximize P(w|s) we have to maximize only P(s|w)P(w). The first of these terms P(s|w) is the probability of typing string s when word w was intended, it is called the *a posteriori* probability or channel model. P(w) is the probability that a writer will type w from among all the dictionary words. It is called source-model or language-model. Many efforts have been made and many different techniques have been devised to model the source and channel behavior. Mainly transition and confusion probabilities are exploited.

Transition probabilities are defined as "the probability that a given letter (or letters sequence) will be followed by another letter (or letter sequence)", these probabilities can be considered as representative of the source or language model, because they try to model the behavior of languages. Language is taken as a Markov process in which adjacent stages (letters in this case) are interdependent. But the probabilistic model of language can never be a substitute for dictionary. [Kukich 1992]

Confusion probability is the probability of confusing a letter (sequence of letters) with some other letter (sequence of letters). These probabilities depend on the process which has caused confusion, e.g. typing or manipulating scanned input (OCR). Since these probabilities represent the behavior of source of error, they are called channel probabilities. [Kukich 1992]

Bayesian classification was first applied for OCR errors correction in 1959 by Bledsoe. Bledsoe used log probabilities in order to simplify calculations. He ignored the language model and only developed a channel model. His channel model was based on letter-to-letter confusion probabilities. Since he calculated P(w|s) for all dictionary words, the growth rate of time utilization was linear with the size of dictionary which resulted in poor efficiency for large dictionaries.

[Hanson et. al. 1976] used both confusion and transition probabilities for error correction with out using a dictionary, but the results showed that depending solely on transition probability to represent a language without taking advantage of dictionary results in suggesting those high probability strings as corrections which are not valid words in the language.

Spell checking applications have never relied on transition probabilities to represent language model. Noisy channel approach was first applied on spell checking in 1990 by

[Kernighan et. al.1991]. They used probabilistic approach only for ranking correction. For candidate generation they applied Demarau's single error technique in reverse. They generated all those string, which could be converted to misspelled string by single insertion deletion substitution or transposition. The strings thus generated, were searched in the dictionary and strings which were not found in the dictionary were discarded. The remaining strings were considered candidate corrections. These strings were ranked using noisy channel probabilities. Prior probability P(w) of a candidate word was calculated on the basis of its frequency in a large corpus according to the following formula

$$P(w)=freq(w)+0.5/M+V*0.5 \quad (3)$$

Here freq(w) is the number of times the word w appeared in corpus of size M. 0.5 is added to the frequency count to cater the prior probability of those correct dictionary words whose frequency in the corpus was 0. To normalize the effect of this addition V*0.5 is added to the denominator where V is total number of such zero frequency words.

Channel behavior was modeled using letter-to-letter confusion probabilities. Four confusion matrices were generated for four basic editing operations; del for deletion, ins for insertion, sub for substitution and trans for transposition. Each matrix was of size N*N where N was the language alphabet size. The ij^{th} (i^{th} row j^{th} column) entry in del matrix was the number of times that i^{th} alphabet was deleted when followed by j^{th} alphabet. The ij^{th} entry in ins matrix was the number of times that i^{th} alphabet was inserted after j^{th} alphabet. The ij^{th} entry in sub was the number of times i^{th} alphabet was substituted by j^{th}. The ij^{th} entry in trans matrix was the number of times that i^{th} alphabet was transposed by j^{th} alphabet when i^{th} alphabet was followed by j^{th} alphabet.

These matrices were populated using a training data set. Conditional probabilities were computed by dividing these confusion frequencies by the total number of times i^{th} character or the sequence of i^{th} and j^{th} characters appeared in training set.

Conditional probability P(w/s), where s is input string and w is any dictionary word can be computed using following formula. (w_p and s_p are p^{th} characters of string s and w respectively.)

$P(w/s)$	=	$del[w_{p-1},w_p] / Count[w_{p-1}w_p]$	if deletion
$P(w/s)$	=	$ins[w_{p-1},w_p] / Count[w_{p-1}]$	if insertion
$P(w/s)$	=	$sub[s_p,w_p] / Count[w_p]$	if substitution

P(w/s) = trans[w_p,w_{p+1}] / Count[$w_p w_{p+1}$] if transposition

The model was first initialized with uniform probabilities and then was iteratively run on training set updating the confusion frequencies in each iteration. This method of training a model is an instance of EM algorithm in which model parameters are iteratively re-estimated until they reach some stable state. [Jrafsky 2000]

This technique was tested on a set of errors each having two potential corrections, the results were compared with selections made by three human judges (considering majority vote). 87% of the time the spellchecker and human judges agreed on the same choice.

The limitation of this technique is that it does not deal with multiple errors. Although 80% of misspellings are due to single errors but the remaining 20% error should not be totally ignored.

A much more generic and sophisticated technique utilizing noisy channel model was proposed by [Brill and Moore 2000].

The technique proposed by [Brill & Moore] relies on probabilistic model to generate candidates; ranking of candidates is implicit. Again confusion probabilities are exploited to model channel behavior, but the editing operations for which these probabilities are calculated are much more generic, instead of using letter to letter editing operation like [Church & Gale1990], string to string editing operations are used, where string can be a sequence of zero or more characters. Use of generic operations makes the model capable of handling multiple errors as well as single errors.

Let $\sum$ be the set of input alphabet the model allows editing operations of the form

$\alpha \rightarrow \beta$ or $P(\alpha|\beta)$
Where $\alpha, \beta \in \sum^*$

To compute P(s|w), both s and w are divided into partitions r_1, r_2, r_3…

$$P(s|w)=P(r_{1s}|r_{1w})*P(r_{2s}|r_{2w})*P(r_{3s}|r_{3w})\ldots. \quad (4)$$

For example

P(معیزاموئیز)=P(موام)*P(ع اء)*P(یز ا یز).

A word can be partitioned in many different ways therefore all-possible partitions are applied and only the maximum probability value is considered.

P(α→β) can be conditioned on position of α and β in the word i.e. P(α→β|PSN) where PSN can be start, middle or end. This increases the accuracy because certain errors are more probable on certain positions in the word. For example, in Urdu, P(ہ→|) is much greater at word ending position as compared to start or middle.

[Brill & Moore] trained the model to set P(α→β) parameters. For training they used a set of Error-Correction pairs. For each s_i, w_i pair of Error and Correction, s_i and w_i are first aligned with each other such that the alignment minimizes the editing distance between the two. This alignment gives us non-match α→β substitutions. In order to incorporate contextual information, different forms of α→β substitution are considered each time including N additional character from the left or right context of α or β.

Once the number of all α→β substitution in training data are counted, P(α→β) can eb found using following formula

$$P(\alpha \rightarrow \beta) = \text{count}(\alpha \rightarrow \beta)/\text{count}(\alpha) \quad (5)$$

As we have seen, count (α→β) can easily be found using training data. But finding count(α) is a bit hard. If we train the model using a corpus we can simply count the number of occurrences of string in the corpus to find count(α), but if count(α→β) is computed using training set then we have to use some independent corpus to estimate count(α), this can be done by counting the number of occurrences of in that corpus and then normalizing it by the rate at which human beings make spelling mistakes.

All α and β parameters were compiled as trie* of tries. A big trie represented all α strings and all those nodes of the trie at which any of the α strings ended pointed to another β trie representing all those β strings that could be confused for string α.

In the original technique as proposed by [Brill and Moore] the dictionary was precompiled into a trie. For each input word they traversed the trie while computing

* Trie is a special kind of tree in which the number of children of every node is equal to the size of language alphabet.

editing distance at each node between the input string and the prefix string ending at that node. The weights for editing operations were computed on the basis of α→β confusion probabilities, which were acquired by searching α and β tries in parallel with the dictionary trie.

α and β were stored in reverse order. This helped traversing them in parallel with dictionary trie traversal. At each node in dictionary α trie is traversed from root downwards while traversing the dictionary upwards and matching the nodes in both tries, if an end node is reached in a trie then starting at that point input string is matched with corresponding β trie.

A trigram language model was used. The probability of occurrence of a word was decided on the basis of the two words to the left of the erroneous word, since the spellchecker is invoked immediately after typing of each word, only left context of the word is available. Using trigram language model gave better result than not using any Language Model. Especially one-best accuracy increased significantly. Church & Gale reported an accuracy of 98.8%, using positional confusion parameters with a context window of 3.

Improved Noisy Channel Model

In an improved noisy channel model, proposed by Toutanova & Moore (2002), α and β were sequences of phones. In this technique all dictionary word and the misspelled string must first be converted from letter sequences to phone sequences. It is easy for dictionary words because their pronunciation is known but for misspelled word some letter to phone translation model is required.

The most commonly used letter to phone model is Fisher's N-Grams model [Fisher 1999]. In this model a probability is assigned to each letter to phone conversion on the basis of its left and right context. Rules are written for this probability assignment. General form of these rules is:

[Lm.T.Rn→ph1p1.ph2p2] (6)

This shows that the letter T, with a particular sequence of m letter at left and n letters at right is pronounced as phone ph1 with probability p1 and ph2 with a probability p2. [Fisher 1999] [Toutanova & Moore 2002]
An example of the rule is:

[کوی حرف ـ ن ـ س]→[__ .6, ن .4]

This rule says that 60% of the times ن becomes silent when it is followed by س word medially.

This 'phone sequence' to 'phone sequence' comparison model was called pronunciation modeling by Toutanova and Moore (2002). In this pronunciation dependent noisy channel model, we first select the most probable pronunciation for the erroneous word and then select that word from dictionary whose pronunciation is closest to erroneous word's pronunciation.

To compute P(s|w) we compute P(pron_s|pron_w)*P(pron_s|s). If more than one pronunciations are generated for a misspelling, only that pronunciation is considered which gives maximum probability value. If the dictionary word also has more than one valid pronunciations then probability value is averaged over total number of valid pronunciations.

They used this model in combination with the original letter based model. The probability $P_{PHL}(w|s)$ from the phone based model was combined with the probability $P_{LTR}(w|s)$ from the letter based model to get combined score $S_{CMB}(w|s)$ using following formula,

$$S_{CMB}(w|s) = P_{LTR}(w|s) + \lambda P_{PHL}(w|s) \quad (7)$$

Parameter λ was set during training to maximize accuracy rate.

Toutanova and Moore showed that a combination of letter based and phone based noisy channel models, shows better performance as compared to individual performances of any of the two models. The combined approach exactly guessed the correct word for 95.58% errors. Three-best accuracy reached 99.50%.

2.2.1.2 Context-Based Error Correction

There are many application in which auto correction is a requirement, for example NLP applications, speech synthesis and OCR etc. in these applications human intervention is not possible therefore spellchecker can not suggest more than one corrections. It was discovered by Church and Gale (1990) that the best correction suggested without using context information, might not actually be the best solution. For example the noisy channel spelling correction model, which detected corrections using reverse single-error method and ranked them according to probability scores, suggested 'acres' as the most probable correction of 'acress ', but if we consider the context,

… was called a "stellar and versatile acress whose combination of sass and glamour has defined her … [Kernighan et. al. 1990]

We can clearly see that actress was the actual word. This observation led them to the idea of using context information, at least for ranking purpose. Church and Gale were the first to use context information for spell checking.

Keeping the ranking issue aside, there is even more serious issue of real-word spelling errors that lies entirely out of the range of non-word spelling correction techniques. According to some studies quoted by [Kukich 1992] 15 to 40% of total spelling errors are real word errors. Again these errors pose more serious problem in the applications where auto correction is required, the need of solving this problem was first realized for NLP applications.

Unlike Non-Word errors, for Real-Word errors, detection is a much harder problem compared to correction. Because the error is not actually a lexicographic or spelling error, (e.g. form→from, the→he) rather it is some syntactic, semantic or some times pragmatic disagreement.

NLP applications viewed real-word spelling errors as violation of NLP constraints, there are generally five types of constraints in NLP, Lexical, Syntactic, Semantic, Discourse and Pragmatic. Lexical errors are addressed through isolated-word error correction; among the other four types of constraints, a greater number of real word errors result in violation of syntactic constraints. Due to this fact, and also due to the relatively easily manageable nature of syntactic category constraints, NLP applications focused on syntactic rules to resolve ambiguities created by real word spelling errors.

In one approach used in NLP, the vary rules that capture the error are used to correct the error. Combining syntactic information and the semantic concept that we have reached, highlights the problem area and helps reaching possible solutions.

One way of merging syntactic knowledge and semantic concept is the expectation-based techniques. In these techniques at each step during processing of input, system builds a list of all those terms that are expected to be input next. This list is built on the basis of syntactic and semantic knowledge. If the next input is not one among the list of expected terms then it can be a potential error. This technique was implemented in a parser CASPER. Slot frame structure was used in which each syntactic or semantic structure is represented as a frame consisting of different slots. These slots have specific order and can be filled by specific types of words. For example frame structure for a predicate transitive verb requires a subject and an object, both of which ought to be nouns and the subject should most likely be an animate one. CASPER maintained the list of expected input terms for the next slot on the basis of syntactic and semantic knowledge of the slots filled so far. Every new input term either invokes a new frame or fills a slot in an existing frame. If any of the slots is left empty or filled wrongly then the syntactic or semantic constraint is violated. [Kukich 1992]

Such techniques, in which semantic knowledge is also used for detecting constraint violation, require immense research in the language.

Statistical techniques are also exploited in real word error correction. In order to capture collocation trends, word bigram and trigram probabilities are used. Since there are millions of words in a language and number of possible combinations even for word bigrams, goes beyond computational limits. A relatively less computationally heavy approach is to use POS bigrams and trigrams. In this approach the words of the language are finely categorized into many POS categories and bigram probabilities for these POS categories are modeled. Now if number of low probability bigrams in a sequence goes beyond some pre specified threshold value, then those bigarm of the sentence are identified as problem area or potential real word error. Once the error is identified, such corrections for the error are suggested that maximize the POS probability and are lexically similar to the error. In fact the correction process is similar to the one used in noisy channel model, except the POS bigrams are used for language modeling.

Though this technique is computationally manageable but it has the disadvantage of not being able to capture those real word errors that do not violate POS constraints and can be identified only with the help of semantic knowledge.

In Another approach proposed by [Tong], the confusion sets of all dictionary words are first identified (confusion set is the set of the words all whose words can be confused for each other) then whenever any of the words from a confusion set occurs in the text, the probability of occurrence of all the members of the set is calculated on the basis of k number of context words on both sides of the actual word. If the probability of actual word turns out to be very small and some other words has very high likelihood of occurrence in that context, then the actual word is considered potential error and more likely words from confusion set are presented as corrections.

All real word error correction techniques either require matured knowledge of the syntax of the language or extensive balanced corpus of the language. The languages, for which neither of the two is available, cannot reach the goal of real word error correction.

3 Urdu Spell Checking

Today thinking of a text editor not having spellchecker functionality seems strange but amazingly none of the Urdu text editors currently available in market provide full-fledged spell checking functionality. Some text editors provide spellcheckers with error detection facility only (PageComposer and Inpage are among them), however no spelling error corrector of Urdu is available yet.

The unavailability of Urdu spellcheckers is mainly for two reasons, 1. No machine-readable Urdu Lexicon is available. 2. As already mentioned Urdu spell checking poses certain problems, which make existing spell checking algorithms inappropriate for Urdu.

In the next section some issues regarding Urdu spell checking are discussed.

3.1 Issues in Urdu Spell Checking

3.1.1 Urdu Diction Problems

The primary function of a spellchecker is to detect spelling errors. In other words, it has to rejects those words that are not allowed in the language. This can be done only if the boundaries of 'what is allowed and what not' are clearly defined. Unfortunately this is not the case with Urdu. There is a great number of such words for which spelling variations are very common. Even the dictionaries do not provide one spelling for such words. As a general trend, dictionaries quote all variations of a word's spelling with out giving any explanation.

Following are given some examples of spelling variations.

There are many such words in Urdu borrowed from Arabic, which end in "ٰی", for example موسیٰ اعلیٰ ادنیٰ مولیٰ. Phonetically these words are pronounced as if the ending letter is "ا" instead of "ٰی". Due to this difference in pronunciation and transcription, it has become a common trend to write these words with an ending "ا". Like اعلا ادنا مولا. As a result, now both styles of writing are being used and accepted.

Some words, borrowed from Arabic or Persian, have an ending "ہ" but this is pronounced as "ا" while speaking. Examples of these words are کعبہ شگفتہ. as a result people have started writing those Hindi and Urdu words that end in "ا", with an ending "ہ". For example کمرہ معمہ تماشہ ناشتہ This variation has also made its way in literature and dictionaries. [Khan 1998]

Urdu character "ء" is some times used in the place of "ی" or in combination with "و". As in چھڑکاؤ پھیلاؤ لئے الاؤنس. Many variations are found in the spellings of such words. Like چھڑکاو پھیلاو لیے الاونس.

"ۃ" was not initially a character in Urdu alphabets, but many words borrowed from Arabic have a "ۃ" at the end. In Urdu these words are written in two ways, with "ۃ" and "ت". Like زکوٰۃ زکوٰت [Khan 1998]

"ز" and "ذ" are homophone characters and are very frequently confused with each other as a result of this confusion, spelling of many words are almost permanently distorted, for example "پذیر" is very commonly written, even in news papers and books, with "ز", while its actual spellings are with "ذ". [Khan 1998]

"ں" is not a separate sound phonetically, it is used to just represent the nasalization of other sounds, logically it should be written adjacent to (either preceding or following) the character whose sound it has to nasalize. And so it is done in words like "منہ" and "مینہ", where its position is just after the letter being nasalized. But in many other words it is either missing or wrongly placed like in "منہدی", "چھانوں" and "گانوں". Four different spellings exist for such words, the spelling variation of each of these is given below.

چھانوں	چھانو	چھاؤں	چھاوں	
گانوں	گانو	گاؤں	گاوں	[Khan 1998]

In Urdu "ُ"(Pesh) represents a short vowel and "و" represents the corresponding long vowel. But in some cases this short vowel is represented by "و" in orthography, for example in "دوکان". Due to this difference in pronunciation and transcription more than one spellings of such words exist. [Khan 1998]

One thing is common in all these spelling variation problems; they have all arisen due to disagreement between pronunciation and spelling. It is a general trend that when languages evolve they tend to become more and more simpl and regular, so in authors view the solution to these problem should reduce the distance between pronunciation and orthography as mush as possible, but those differences of spelling and pronunciation that are very well known and therefore do not cause confusion should not be removed, because making changes in generally accepted forms of words will just increase the confusion rather than decreasing it.

Another reason for confusion is the large number of homophone characters in Urdu; suggestions of merging all homophone characters of a class into one were made [Khan 1998]. But doing so will also increase confusion. A better solution is to accept common variations of a word's spelling, since spelling variations is a normal phenomenon in languages and many languages including English exhibit this behavior. But doing this will not eliminate the need of standardization, because a distinction must be made between variant forms of a word and the generally made spelling errors of a word.

3.1.2 Word Boundary Problem

The input of a spell checker is words. When a document is to be spell checked it is tokenized in order to separate words. This tokenization is generally done on spaces, because spaces represent word boundary. But for the languages like Urdu that use Arabic script, identifying words is not just a matter of tokenizing words on spaces, because some times spaces are inserted in the middle of the word just to prevent the joining of two characters with in a word that are supposed to be separate.

بذلہ سنجی گردن زنی

The reverse situation i.e. not inserting space between two separate words is even more frequent. Because in Urdu writing there is actually no gap between two words, separate words are just not joined with each other. When typing, two words are not joined if the last character of first word is a non-joining character, even if no space is inserted between

the two words. In such situations it is very common trend to not insert space, because visually it makes no difference. But the tokenizer will consider the two words as one and the spellchecker will flag it as spelling error.

اپنا گھر میز پر

One possible solution for this problem is to use two types of spaces one for keeping the letters from joining, and the other will act as word separator. This will solve the problem of space insertion with in word boundary, but the problem of not inserting space on the word boundary will remain there.

3.1.3 Diacritics Problem

Diacritics are optional in Urdu, if we write a word without diacritics or with partial diacritics it is perfectly alright and not considered a spelling error, but if we write a word with wrong diacritics, it is an error.

In fact diacritics are used to help understanding the pronunciation of a word, because with out diacritics we cannot predict short vowel (in Urdu diacritics represent short vowels), as they are not represented by alphabet characters.

Though this situation is not very complicated but it may pose some problems while spell checking. If the spell checker completely ignores the diacritics, it will not be able to capture wrong diacritics errors. If we consider diacritics and alphabets at same level, then the difference measure between two strings differing only in one diacritic symbol will be equal to the difference measure between two strings differing in one alphabet. And this, we know, should not happen. This problem can be handled by considering diacritics at some different and relatively lower level of importance compared to alphabet. Doing so will be safe because diacritics and alphabets are rarely confused with each other with the exception of "ُ" (Pesh) whose sound is sometimes represented by "و" as in "دکان". Such exceptions can either be handled separately or can simply be ignored.

4 Problem Statement

The statement of problem for the thesis work presented in this document is:

> *"Given a list of valid Urdu words and an erroneous word to be spell checked find a word from the list which is most likely to be the actual intended word."*

Providing the list of valid Urdu words and designing algorithm for detection of errors is not in the scope of the work.

5 Methodology

Methodology adopted for development of spelling correction algorithm for Urdu included following major steps

1. Study of spelling error trends in Urdu
2. Testing existing spell checking techniques on Urdu
3. Optimizing the techniques for Urdu
4. Developing solution for Urdu specific spell checking problems
5. Combining the techniques

In order to see whether existing algorithms can be effectively used for error correction in Urdu, a study was performed to analyze spelling error trends of Urdu. From this study factor causing spelling errors were identified and also the applicability of already identified errors trends on Urdu was verified. During this study Urdu specific spell checking problems were also discovered.

After studying the spelling error trends of Urdu next step was to evaluate the effectiveness of existing spell checking techniques on Urdu and modifying them to address Urdu specific problems.

The techniques selected for testing include Single Edit Distance technique and Soundex. The idea was to use both in combination expecting that single edit techniques will correct typographic errors and Soundex will correct phonetics based multiple errors. Single edit distance was preferred over multiple edit distance due to its efficiency. If we apply single edit in reverse we have to check 73n+35 (with alphabet size 36 and word size n) suggestions for validity. As we increase the edit distance from single to double, triple and so forth this number gets multiplied with it self. Even for multiple edit distance with threshold 2 it becomes $5329n^2$ making the technique very inefficient and useless for all practical purposes. An alternative can be to use the famous dynamic programming algorithm for calculating Edit distance [see Appendix A] between error and every dictionary word but this will also require $N*n^2$ time where N is the dictionary size again making it very inefficient. Another alternative can be to use some tree like efficient in-memory data structure (for example automaton of the entire dictionary) for keeping dictionary and traverse this data structure while computing edit distance between error and the dictionary strings. This becomes relatively efficient because pruning techniques can be used to avoid irrelevant distance measures. But for large dictionaries the size of automaton becomes unmanageable.

In 'Spelling error correction' section many techniques of spelling correction other than the above ones are presented, two of these techniques: N-Gram Similarity measure and Noisy channel model, are important enough to deserve an explanation for not including them in the experiment.

The N-Gram based techniques aren't used because they don't perform well on transposition and substitution errors and all studies on spelling errors (including author's study of Urdu spelling errors) show that transposition and insertion together make nearly 55% of the total spelling errors. Using N-Gram based techniques simply means that either these 55% errors are ignored or the N-Gram similarity threshold is kept so high that too many irrelevant results are obtained. On the other hand if Single Edit Distance approach is used, it ensures that none of the basic four types of errors is ignored.

Noisy channel model is a probabilistic model. It assigns probabilistic weights to editing operations. This model when trained on large text of a language automatically sets its weights according to the error patterns of the language. As already mentioned the major factor affecting these error patterns is the keyboard layout. [Kukish 1992] reported that 58% of typing errors are due to adjacent keyboard keys. Unluckily there is no standard keyboard layout for Urdu typing; all text editors provide their own keyboard layouts, which are quite different from each other [see Appendix B.]. As a result the Urdu corpus gathered from different sources was typed using different keyboards and therefore could not provide useful data for training. In this situation it seemed a better idea to use deterministic techniques rather than using probabilistic ones. Partly for this reason and partly for the fact that intelligent combination of deterministic techniques (which generally tend to be more efficient and less complex) performs almost as well as does the noisy channel model [Hodge 2003], it was decided that Noisy channel model not be used.

Test Data

The selected techniques were tested on a test data of 744 errors correction pairs. These errors were selected randomly by spell checking text from different sources[*]. Corrections for theses errors were specified manually, and if there was some ambiguity or confusion in identifying the correction for some error, the error was looked up in the source text and was corrected on the basis of context information. During the selection of test data one

[*] Online Jang Newspaper, www.jang.org/urdu
Text from initial draft of an online dictionary being developed at CRULP.
Text from a corpus developed at CRULP gathered from different sources (books from Ferozesons and Iqbal Academy).

thing was kept in mind that it should contain appropriate ratio of all types of errors. Table-6. provides the profile of the test data.

	Number of Errors	%age of Total Errors
Insertion	54	7.25%
Deletion	71	9.54%
Substitution	121	16.26%
Transposition	25	3.36%
Space Deletion	443	59.54%
Space Insertion	20	2.68%
Multiple Errors	10	1.3%
Total	744	

Table 6. Error profile of test data

The number of space deletion errors included in test data is less than their actual proportion in Urdu typing errors. This was done so because handling some space deletion errors means handling them all and 443 is already a very big sample size from a sample space with little deviation.

When the techniques were tested independently only relavent subsets of this test set were included i.e. when testing technique for 'Space deletion errors correction' only space deletion errore (443 in number) were used as test data similarly when 'Single Edit' and 'Soundex' techniques were tested, only non-space errors (280 in number) were used as test data. But for testing the combined approach whole set of test data was included.

WordList

The Word List used for spell checking was the Lexicon prepared at CRULP* containing a total of 112481 words. The words whose corrections were not present in the lexicon were not included in the test data. The lexicon contained all forms of word therefore no morphological processing was involved.

Spellchecker application

The spellchecker application was made so that it could input text in form of error correction pairs. When provided with an input file containing error correction pairs it

* CRULP: Center of Research in Urdu Langue Processing, FAST-NU

finds corrections for every error from the lexicon then it ranks the corrections and outputs the correction list and the rank of actual correction in this list in an out put file.

In the case of space insertion errors the error spans more than one word, in this situation error correction pairs cannot be used for testing. Therefore space insertion errors were spell checked in carrier sentences.

5.1 Study of Spelling Error Trends in Urdu

A study was performed to identify error patterns in Urdu. The data used for the study was gathered from three different sources

1. Urdu Newspapers
2. Urdu term papers typed by graduate and undergraduate university students
3. A corpus of Urdu Text (1.7 million words)

The methodology used for study of the data from first two sources was different from the one used for analysis of the data from third source.

In the first case, the data were available in the form of hard copies and were manually spell checked. In the second case the corpus was first tokenized to separate words then frequency analysis of the words was done. On the basis of this frequency analysis, those words were selected for study which appeared only once in the whole corpus, considering them potential errors. It was manually verified whether these low frequency words were actually errors or not and only those were included in study that turned out to be actual errors.

Due to the difference in the nature of data and in the methodology of study, results for the two studies are discussed separately.

5.1.1 Study 1.

First we analyze our results obtained from the study of data from first two sources. Results of the study are shown in Table 2 & Table 3.

	Typographic	Cognitive
Students	57.69%	42.31%
Newspapers	91.18%	8.82%
Overall	74.44%	25.57%

Table 2. Comparison of ratios of typographic and cognitive errors

The statistics from both sources are separately entered in Table 2 due to clear difference in the nature of mistakes in the data from the two sources. The data from newspapers mostly contains typographic errors with a small fraction of cognitive errors on the other hand the ratio of cognitive errors in the students' data is much higher (about 42%). (Here those errors are being considered cognitive for which the pronunciation of misspelled

word was similar to the intended word). It seems that the newspapers' data was a little too clean for the study; (perhaps due to proof reading) the general trends of errors might actually be different than indicated by it.

The results of the study agree with the studies conducted by Damerau (1964) and Peterson (1986). Table 3 contains data about four basic types of errors specified by Damerau, two additional categories space insertion and space deletion are added.

	Number	% Age
Substitution	101	47.42%
Deletion	56	26.29%
Insertion	28	13.15%
Transposition	17	7.98%
Space insertion	4	1.88%
Space deletion	7	3.29%
Total	213	92.61%
Total number of errors was 230.		

Table 3. Single edit distance errors in Urdu.

Substitution errors have the highest frequency. This can be attributed to the fact that in Urdu most of the phonetics based cognitive mistakes are single letter substitutions; therefore the substitution frequency is in fact a sum of typographic and phonetic substitutions frequencies.

There are a couple of factors other than adjacent key substitution that can account for substitution errors in Urdu.

The first is that many letters in Urdu are typed with the ***shift key*** pressed (due to greater number of alphabets, 39 in total), so a letter might be replaced with another letter if same key is used for typing both of them.

For example the data from university students was typed using Microsoft keyboard layout in which 'ج' & 'چ', 'ن' & 'ں', 'ح' & 'خ' have same keys [Appendix B] and the interchange of these letter pairs was more frequent.

The second factor is ***shape of letters***. The letters having similar shapes are confused for each other. In 36 out of 101 substitutions, the shape of substituted letter was similar to the

original letter. For example ‘ج’ is confused with ‘چ’. ‘ٹ’ is confused with ‘ت’. Shape plays role in generation of spelling mistakes for two reasons. First, if a substiturion is shape based there is less chance of it being caught by the typist due to visual similarity, Second, in the case of ‘Look and Type’ typing like for newspapers or other publications, books, journals etc., the person typing is not the writer of the text and the text is provided in hand written form, in such situation there is a good chance of making shape based errors. Our study included both types of data and both contained shape based substitutions.

26 out of a total of 230 errors could be considered cognitive or phonetics based. The most common of these was interchanging ‘ز’ & ’ذ’ and ‘ض’ & ‘ظ’.

For example confusing ‘پذیر’ with ‘پزیر’ and ‘لحاظ’ with ‘لحاض’.

These errors are mainly caused due to ***Homophone Characters***. Homophone characters are those characters, which represent the same sound. In Urdu the number of Homophone characters is relatively grater compared to English. Following are listed the homophone character sets of Urdu.

ز،ذ،ض،ظ

ص،س،ث

ط،ت

ک،ق

ا،ع

ح،ہ،ھ

Among the 17 mistakes that were not at an edit distance of one from the correct word only 6 could be tackled phonetically. Viewing the matter the other way, we may say that one fifth of the 25% cognitive errors are such that they cannot be detected using single edit distance techniques.

It was also observed that in Urdu word initial errors are as common as are word medial or final errors. Especially word initial omission errors (مجھے → جھے, اسلامی → سلامی) are very common. Moreover phonetics based substitution errors (زینت → ذینت , زیب→ ذیب) are as common word initially as they are word medially.

Space insertion & space deletion together account for 5% of the typing errors. The errors of this type are more frequent in Urdu because in manual writing Urdu writer are not habitual of putting spaces between words.

Another interesting observation regarding these errors was that 25% of these were real word errors i.e. they resulted in valid Urdu words. For example:

سلامی → اسلامی

ماہین → ماہرین

ترکی→ترقی

چھوٹ→جھوٹ

5.1.2 Study 2.

Now we come to the results of corpus data study, which are presented in Table 4 and Table 5.

Deletion	43	17.99%
Insertion	49	20.50%
Substitution	109	45.61%
Transposition	17	7.11%
Diction Variation	21	8.79%
Total (non space errors)	239	24.51%

Table 4. Results of Corpus Data Study

Non space errors	239	24.51%
Space Related Errors	736	75.49%
Total Errors	975	

Table 5. Comparison of the space related errors with general errors.

Statistics regarding four basic types of errors are again in agreement with previous studies, but the major difference is the large number of errors due to space insertion or deletion. This type of errors could not be captured through the manual study since such mistakes most of the time make no change in the visual form of the word, while for error analysis of corpus, the corpus was first programmatically tokenized on spaces and punctuation marks in order to separate words. Due to inappropriate use of space, too many run-on and split-up words were found, 75% of the total 975 Non-Word errors was due to missing or wrongly inserted space. Space deletion is much more frequent compared to space insertion; perhaps because we always want to minimize typing effort therefore spaces are omitted intentionally but inserted only mistakenly. This type of errors are not actually mistakes, the user intentionally omits space knowing that it will create no difference in the visual form of the word. These mistakes are not a problem for the reader (the difference in statistics of the two studies is a proof for this implication); they are just a problem for spell checker. If this problem is not properly tackled the spell checker will give too many false alarms.

From this study we can conclude that spelling errors in Urdu are less phonetic moreover they also depend on shift key and the shape of alphabets. We have also seen that space insertion and deletion errors can pose serious problems while spell checking.

5.2 Techniques employed for Error Correction

5.2.1 Soundex

Soundex algorithm is one of the techniques employed for spelling correction. Two variations on original Soundex algorithm were tested. These are:

1. First letter code assignment

Unlike original Soundex algorithm, code was assigned to first letter of the word. This variation was introduced on the basis of the study of error patterns in Urdu. During this study it was observed that in Urdu spelling errors at first position are as common as are on any other position in the word. This is probably due to big number of homophone consonants sets in Urdu. Therefore first letter was encoded with expectation that it would result in better recall percentage. Another advantage was also gained by encoding first letter; it reduced the total number of codes thus making the index more manageable.

2. Increased number of alphabet groups

Another variation on the basic algorithm was the number of alphabet groups to which codes were assigned. In Urdu total number of letters in the alphabet is 43, dividing them into 10 groups assigning codes 0-9 results in combining irrelevant characters into one group. To avoid this problem Hexadecimal codes were used, the idea was taken from Hodge et. al. (2003). Using hexadecimal codes we have 16 letter groups giving enough freedom to not combine irrelevant characters in same group. But as the number of groups increases the number of suggestions decreases. Though the excluded suggestion are only those that were there due to irrelevant combinations of characters but both variations of the techniques i.e. Soundex 0-9 and Soundex 0-F were tested for the sake of comparison.

All combinations of the above two variations were tested making a total of four Soundex variants.

No changes other than the above mentioned ones were made in the basic Soundex algorithm. Code length was 4. Only a few N-Gram substitutions were made, because in Urdu phonetics based letter groups substitutions are rare. Vowels were ignored during code assignment. In Urdu identifying long vowels is a little tricky, since same alphabets ا, و and ی that are used as long vowels are also used as consonants in certain contexts. Actually they behave as consonants when they occur word initially or syllable initially. Hence to identify these consonants syllabification was required, but in Urdu a word can be syllabified only if it is marked with diacritics and generally words are not marked with

diacritics. Therefore only a rough guess could be made about the situation of these long vowels. To do this and also to perform other N-Gram substitutions, the string was normalized before code assignment.

Normalization

Following normalization rules are applied:

1. Treatment of ا

Word initial ا was converted to ع, because word initially ا is some times confused with ع. For example confusing عجلت with اجلت or عرق for ارق.

2. Treatment of آ

آ was converted to عا.

آم → عام

3. Treatment of ہ

Word final ہ was converted to ا. Actually in Urdu word final ہ is always pronounced like ا. For example see the sound of ہ in following words: ناشتہ،کمرہ،اہلیہ

4. Treatment of ھ

In Urdu ھ represents the aspiration of the sound preceding it. During normalization, if the preceding character is among the ones, which can not be aspirated in Urdu like ا, س etc. , ھ is converted to ہ , assuming that actually the sound of ہ was intended, otherwise ھ is ignored like vowels.

For example if ھ is preceded by ا in some erroneous word, it is very likely that actual intended character was ہ. Consider the common errors of writing اہمیت as اھمیت or writing لاہور as لااھور. Some rarely aspirated characters like ل were not included among

aspirated alphabets. Table 6 shows the list of the aspirated characters for which ھ is not converted to ہ

چ	ج	ٹ	ت	پ	ب
	گ	ک			

.Table 6. List of characters for which ھ is not converted to ہ

5. Treatment of ی

ی when occurs word initially or is surrounded by ا or/and و (i.e. occurs in intervocalic position) is converted to ژ. Actually in these positions ی behaves like consonants. Consider the examples of یار, یوں, لایا and سویا.

6. Treatment of و

و when occurs word initially or is followed by ا or ی it remains as is, otherwise it is converted to ہ (pesh) which indicates its vowel behavior during code assignment. Consider the difference in sound of و in سویرا, جواب and سورج.

The last three rules are mutually dependent therefore they were applied in two passes. In first pass behavior of ی and ہ was established as vowel or consonant. In the second pass the behavior of و was decided on the basis of following character, if the following character was vowel i.e. either ا or the vowels identified in the first pass, then it was considered consonant otherwise it was considered vowels. Behavior of ی is also reviewed in second pass because due to change in status of some more consonant ی s can be identified.

If this was all done in one pass, then the و in سویا had been considered consonant and ی in سایہ had been considered vowel.

After normalization Soundex code is assigned to the normalized string.
Table 7 shows the Code assignment for Soundex 0-F and Table 8 shows the Code assignment for Soundex 0-9

Code	Alphabets
0	ث س ص ش
1	ت ط ٹ ۃ
2	ز ض ظ ذ
3	ج چ ؔ
4	ہ ح ھ
5	خ ک ق
6	د ڈ
7	ب پ
8	ن ں م
9	گ غ
A	ر ڑ
B	ژ ی
C	ا آ ع
D	ف
E	ل
F	و

Table 7. Soundex codes 0-F

Code	Alphabets
0	ث س ص ش
1	ت ط ٹ ۃ
2	ز ض ظ ذ
3	ج چ ؔ
4	ب پ ف و
5	خ ک ق
6	د ڈ ر ڑ
7	ن ں م
8	گ غ
9	ل

Table 8. Soundex codes 0-9

Ranking

Soundex algorithm does not provide any mechanism for ranking of suggestions but the result set retrieved by Soundex is some time too big to be presented as suggestion list. Therefore some of ranking was required so that only top ten results could selected. The measure used for ranking is the frequency of the word in the language. This frequency

was measured by counting the occurrences of a word in a corpus of 1.7 million words and then dividing it with total number of words in the corpus. This type of weight assignment is also called language modeling, because the frequency of a word represents the probability of its occurrence in any text of the language.

The result set returned by the Soundex algorithm is sorted on frequency and the top ten words are added in the suggestions list.

Results

The four variants of Soundex; Soundex 0-9, Soundex 0-F, Soundex 0-9 with first letter encoded and Soundex 0-F with first letter encoded were tested on test set of 280 errors. These errors did not included space related errors, since the technique is not designed for this type of errors; including them in the test data could only worsen the results.

Table 9 shows a comparison of the results obtained from the four Soundex variations. Soundex 0-F with first letter encoded out performs the rest. Though total percentage recall of Soundex 0-9 is better than the rest but the result set returned is unreasonably big and after frequency ranking, Top-Ten recall rate decreases by 18%, which shows that the result set contained too many irrelevant words which made their way to the top due to high frequency in the language.

An interesting thing shown by the results is that any of the variations alone (encoding first letter or increasing number of codes) perform a little poorer than the original Soundex 0-9-L. only the combination of both performs better. This is so because only increasing the number of codes make the result set very small and only encoding the first letter makes the result set too big, 92 suggestions on average. The combination of both gives a reasonable result set with overall better recall.

	Top One %age	Top Five %age	Top Ten %age	Above Ten %age	Total Recall %age	avg size of result set
Soundex 0-F (with 1st letter encoded)	26.2	44.4	49.5	5.4	54.8	21
Soundex 0-9 (with 1st letter encoded)	20.1	35.5	43.0	17.9	60.9	92
Soundex 0-F-L (with 1st letter not encoded)	23.7	36.2	39.1	0.7	39.8	7
Soundex 0-9-L (with 1st letter not encoded)	22.6	40.9	46.2	3.2	49.5	21

Table 9 Performance results of Soundex and its variants.

This is clear from the performance results of Soundex and its variants that Soundex algorithm alone cannot be used for spell checking, even the best Top Ten recall percentage is nearly 50% meaning that 50% of errors can not be corrected using Soundex. A small advantage was that Soundex recalled some multiple error corrections as well but it was just a small fraction of the total errors.

5.2.2 Single Edit Distance

Single Edit Distance technique was used in reverse i.e. single edit operations were applied on error and the resulted string was tested for validity. Bigram validity test was used for efficiently checking the validity of the string. A bigram binary matrix of size 52*52 was built. The value at i^{th} row and j^{th} column indicates whether i^{th} character followed by j^{th} character forms a valid bigram in Urdu. For all these bigrams the frequency of their occurrence in the Lexicon was computed. It was found that 830 out of total 2025 bigrams of Urdu were invalid i.e. they never occurred in the dictionary. After applying an editing operation on a string, its bigrams were first tested for validity, if the bigrams of the string pass the validity test then the string was searched in the dictionary. To further speed up the process only those bigrams were tested for validity that are affected by the editing operations, in other words all unchanged bigrams are considered valid. This is not just an assumption, actually before applying editing operations the bigrams of erroneous string are tested for validity and if any invalid bigram is found, editing operations are applied only on that bigram.

Single Edit techniques like Soundex does not provide any ranking mechanism so the ranking was done as a separate process.

Ranking

Three factors were exploited for ranking, these are, frequency, sound-similarity and shape-similarity.

Keyboard adjacencies could also be a good parameter for ranking but as already mentioned there is no standard keyboard layout for Urdu and an effort was made to avoid using environment specific factors so that the solution remain generic.

frequency-based ranking

Detail of frequency-based ranking is already given in the ranking section of Soundex. Same technique was used here.

Sound similarity based ranking (Soundex)

Soundex code was used to capture sound similarity between the error and the suggested corrections. To do this Soundex codes were generated for the error and for all corrections and those corrections whose Soundex code matched the Soundex code of error were considered more likely for being actual intended word. In the case of substitution-based correction the Soundex code of only the substituted character was compared with the Soundex code of original character in its place and if the two codes were same the words

were considered phonetically similar. The encoding scheme used was Soundex 0-F with first letter encoded.

If more than one correction matched with the error, further ranking was done on the base frequency.

Shape similarity based ranking (Shapex)

Shape similarity was used as ranking parameter on the basis of the study of error trends in Urdu. During this study it was observed that in 35% of the substitution errors, the shape of the original character and the substituted character is similar. To make use of this fact codes were assigned to alphabets on the basis of shape similarities. These codes are given the name Shapex. Table 10 shows the Shapex codes assigned to the letters of Urdu alphabet. Most of codes are assigned on the basis of similarity in word medial shapes of characters.

Code	Alphabets
0	اآل
1	ب ہ پ ی ئ
2	ت ٹ ث ن س ش
3	ح خ ج چ
4	د ڈ ذ ر ڑ ز ژ و ؤ
5	ص ض
6	ط ظ
7	ع غ ف ق م
8	ک گ
9	ے ئے

Table 10. Shapex Codes

Shapex based ranking was applied only on substitution errors. Moreover, like Soundex based ranking, if more than one corrections matched with the error, further ranking was done on the basis of frequency.

Combining all three ranking approaches

A combination of above-mentioned three ranking approaches was also tested. In the combined approach three levels of ranks were defined.

Rank 1: Both Shapex and Soundex codes matched
Rank 2: Either Shapex or Soundex code matched
Rank 3: Neither Shapex nor Soundex code matched

The corrections in all ranks were independently sorted on frequency. Then top ten corrections were selected such that first all correction from Rank 1 were included and if they were less then 10 then the corrections from Rank 2 were included and so on until the number of correction reached ten of the correction list was exhausted.

Results

Edit Distance algorithm was tested on a set of 280 errors, which did not include space related errors. The algorithm was tested in combination with above-mentioned four ranking approaches. The results of all these approaches are given in Table 11.

	Top One %age	Top Five %age	Top Ten %age	Above %age	Total Recall %age	avg size of result set
SE + FR	58.1	90.0	93.5	1.1	94.6	8.5
SE + SXR + FR	64.2	89.6	93.9	0.7	94.6	8.5
SE + SPR + FR	64.5	89.6	93.5	1.1	94.6	8.5
SE + SXR + SPR + FR	71.7	87.1	93.5	1.1	94.6	8.5

Table 11. Results of Single Edit Techniques with different ranking approaches.

SE: Single Edit
FR: Frequency based Ranking
SXR: SoundeX based Ranking
SPR: ShaPex based Ranking

The results show that a combination of all three ranking approaches gives greater number of first place matches compared to any of the ranking techniques alone. Overall Top-Ten Recall rate is excellent. This is due to greater percentage of single edit errors in Urdu. The results obtained for Soundex only and for Shapex only ranking are almost same. This shows that in Urdu typing errors, shape similarity performs as important role as does the sound similarity.

5.3 Handling of Word Boundary Problem

In Urdu, 75% of the typing errors cross word boundaries. These include space deletion and space insertion errors. This type of errors is handled separately.

5.3.1 Space Deletion Errors

Space deletion errors form above 90% of the total space related errors. The root cause of these errors is the existence of non-joining characters in Urdu alphabets. Actually in manual writing Urdu is written with out spaces between words. During typing one has to insert space between words to keep the words from joining, but if the last character of a word is a non-joining character then space can be omitted since the character won't join anyway.

One way of handling the space deletion problem is to consider space-deletion an editing operation like other editing operations and apply single edit error correction algorithm in reverse to correct the error. This works well but there is a limitation of this approach. The problem is that if more than two such word occur in a sequence that end in a non-joiner character then typist may omit space between all of them. In other word there is a good chance of multiple space deletion errors. Though the error analysis showed that only 5% space deletion errors are multiple errors. But since these errors are actually not errors, just an illusion for spell checker, they should be removed with near 100% accuracy so that they might be auto corrected if required. So the algorithm was modified to handle multiple space deletion errors. The idea was to test all possible partitions of the word for validity, but since space deletion errors result in longer words, checking validation of all partition for such long words could become very costly. To avoid this, only those partitioned were generated and tested for validity that split the word on non-joiner characters' positions. Non joiner characters are listed in Table 12.

ا	د	ڈ	ذ	ر	ڑ
	ز	ژ	و	ئی	ے
	ئے	ؤ	آ	ۃ	ں

Table 12. Non-joiner characters in Urdu.

A recursive algorithm was developed to check all possible partitions of the word. The concept of memoization was used to avoid rechecking the validity of already checked sub partitions. Actual algorithm is given in Appendix C.

During partitioning of words, some joining characters were converted into non-joiner before checking the validity of the partition. These include ن and ی, which were converted to ں and ے respectively. This was done so, because there is a common trend the word that end in end these characters, are joined with the following word. Consider the following examples:

آؤنگا ,آؤنگی ,پانیوالے ,جاؤنگا

Moreover the character آ was considered as non-joiner on both sides. Though there are some exceptions to this rule like بالآخر and مآل, but they are very rare and can be ignored to achieve efficiency.

Ranking

There can be more than one valid partitions of a word. For example برپاکردیا can have three valid partitions:

برپاکردیا برپاکردیا برپا کردیا

Therefore ranking is required to select one best partition. Frequency based ranking was employed for this purpose. Since the corrections in the case of space deletion errors are comprised of more than one word, some mechanism for assigning frequency to this group of words using the individual frequencies of words was required. One simple solution could be to use the average of the frequencies of all constituent words but in this approach some times such partition that contained very rarely used words got their way to top ranks due to other very frequent constituent words. For example there are two valid partitions of آجارہے

آ جا رہے and آ جار ہے

the average frequency of آجارہے was greater than average frequency of آ جا رہے this was due to greater frequency of ہے , but the actual correction is آ جا رہے۔

Another solution, that was actually applied, is to assign the frequency of the least frequent word to the whole partition. This solution worked well. In the above example it ranked آ جا رہے at lower rank due to smaller frequency of the word جا ر.

Frequency based ranking was coupled with subjective ranking; the parameter used for subjection ranking was the number of constituent words in a partition. The greater the number of words lesser likely is the partition for being actual indented sequence of words. This is actually a common sense phenomenon. If a sequence of word can be partitioned in more than one way and our indented sequence is the longer one we will sure put spaces to clarify our intension and remove ambiguity. For example

آ یا جا ,آیا جا

On the top of this another level of ranking was defined; the corrections obtained by partitioning the words on non-joiner characters were ranked higher than the ones obtained by partitioning on joining characters.

So the overall ranking was as follows:

Rank 1: Space Deletion on Non Joiner chars
- →Rank 1.1: with two parts
  - →Rank 1.2: with three parts
  - →Rank 1.3: with four or more parts

Rank 2: Space Deletion on joining characters

If the frequency of a correction in Rank 1 is less than a certain threshold it is moved to Rank 2.

Results

The space deletion error correction mechanism was tested only on space deletion errors because it is a specialized technique and is not designed for other kinds of errors. The test set comprised of 443 space deletion errors. Table 12 shows the result of both Single space deletion correction and Multiple Space Deletion correction.

Multiple space deletion increased the recall rate by 4%. Nearly two percent of the errors are not corrected by the technique; all these errors contained other mistakes along with the space deletion mistake. For example لگوارہا→لگورہا

	Top One %age	Top Two %age	Not Found %age	Total Recall %age	avg size of result set
Single Sp. Deletion	92.3	94.2	5.8	94.2	1.5
Multiple Sp. Deletions	95.7	98.2	1.8	98.2	1.5

Table 13. Results of Space Deletion errors correction technique.

5.3.2 Other space related Errors

Space insertion and space transposition errors together make nearly 7% of the total space related errors. These errors are not intentional like space deletion errors; they are actual mistakes and fall in the category of typographic errors. No instance of space substitution was found during the analysis of errors.

Space insertion and space transposition errors were handled the same way as the other single edit errors were handled. The process of space insertion or space transposition was applied in reverse on the word boundaries and the resulted word was tested for validity. This approach corrected nearly 50% of the space insertion and space transposition errors. The rest of the errors were multiple space insertions or combination of space insertion and space deletion.

5.4 Combining the techniques

After testing the effectiveness of individual techniques and optimizing them for Urdu, a hybrid algorithm for spelling correction was devised. This algorithm included the best versions of the individual techniques.

Inclusion of Space Deletion and Space Insertion Correction algorithms was necessary to cater 75% space related errors. For correction of rest of the 25% non-space errors, Single Edit Distance technique was used in the main. Soundex 0-F was also included to correct multiple edit distance errors that cannot be corrected using Single Edit technique.

Representative from all three techniques were included in the result set of size ten. The main issue in this combined approach was how to rank the corrections obtained from different techniques, relative to each other.

Ranking

Ranking scheme adopted to specify the relative ranks of corrections received from different correction techniques was somewhat similar to the combined ranking approach used in single edit distance technique. Five level of ranking were defined:

Rank 1: Corrections from Rank 1 of Space Deletion Corrections.
Rank 2: Corrections from Rank 1 of Single Edit + Corrections from Rank 2 Space Deletions + Space Insertion + Space Transposition
Rank 3: Corrections from Rank 2 of Single Edit
Rank 4: Corrections from Rank 3 of Single Edit
Rank 5: Corrections from Soundex 0-F

The corrections within each rank were sorted on frequency then from each rank its quota of corrections was put in the suggestions list. Rank 1 and Rank 5 could each contribute at most 2 corrections, rest of the corrections were taken from Single Edit corrections to make a total of 10 suggestions.

Results

The hybrid technique was tested on a test set of 724 errors, which included 444 space related errors and 280 non-space errors. The technique was tested with and without inclusion of Soundex 0-F algorithm. The results are presented in Table 13.

	Top One %age	Top Two %age	Top Five %age	Not Found %age	Total Recall %age	avg size of result set
SE+SPD+SX	82.57	93.91	96.27	3.32	96.68	6
SE+SPD	82.43	93.08	96.13	3.46	96.54	6

Table 13 Results of Hybrid Approach. SE=Single Edit SPD=Space Deletion SX=Soundex 0-F

The results show that the contribution of Soundex 0-F in overall recall rate is negligible, which reinforces our previous observation that Soundex is not a suitable technique for corrections retrieval, though it plays important role in ranking. Over all recall rate is above 96% which is very good and is comparable even with probabilistic correction models of English which reach an overall recall rate of 98% [Toutanova et. al. 2002]. Top one recall percentage is also satisfactory and is better than the top one recall percentage of deterministic spell checking techniques applied on English, though the probabilistic techniques give much better top one recall reaching nearly 95% [Toutanova et. al. 2002].

6 Conclusion

On the basis of the research presented in this document certain facts regarding Urdu spell checking are established. The most important among them is the fact that a spell checking technique for Urdu can be acceptable only if it provides a satisfactory solution for space deletion errors. From satisfactory we mean that the solution should be able to correct these errors with ideally 100% accuracy and should also be confident enough to automatically replace them or alternatively should not flag them as errors. The algorithm presented in this document for space deletion error correction successfully achieve the first goal i.e. it successfully corrects all space deletion errors but the second goal is not fully achieved; the technique some times (2-3 % of the times) suggests a space deletion correction as best match for errors which are actually not space deletion errors. This was mainly because the corpus used for frequency ranking was not well balanced; it contained pretty frequent occurrences of some seldom used Urdu words. For example the number of occurrences of word دھات in the corpus were more than the number of occurrences of word یہاں. If a more balanced corpus is used for frequency analysis, the level of confidence in identification of space deletion errors can increase.

Another important observation is that in Urdu due to smaller word lengths, 5 to 6 characters on average, single errors are more common compared to English. Therefore a technique which addresses only single errors can show reasonably good performance as is clear from the results of the hybrid approach, moreover the errors that are not corrected through single edit technique can also not be corrected though phonetic based techniques in other words multiple errors are not phonetics based. The 4% errors that could not be corrected through the technique suggested in this document are mostly at an edit distance of 2 from correct word. A good thing about these errors is that they do not have many valid words on a distance of one (generally less than 5). This characteristic of multiple errors can be used to further improve the correction mechanism. To increase recall, multiple edit distance technique can be used in combination with single edit techniques but to retain the efficiency of single edit distance techniques the multiple edit distance technique be invoked only when the number of correction obtained from single edit techniques is very small

Another interesting finding is the role of shape similarity in spelling mistakes. During the testing of techniques Shapex was used only for ranking, one can try the use of Shapex for corrections retrieval. Interestingly some of the multiple errors that could not be corrected through Soundex were actually shape based, for example:

کنڑول → کنٹرول　　آتشنردگی → آتشزدگی　　طافوں → لحافوں

Some serious issues regarding lexicon were also encountered. Most compound words were not present in the lexicon. The solution adopted for this problem was that the compound words like space deletion errors were broken into constituent words and then these constituents were tested for validity. But here another problem rose; the constituent words change their form in compounds, and these changed forms were not there in the lexicon. For example consider the following compounds

صائب الرائے

ناممکن الحصول

اردوے معلی

ازروے احتیاط

A better solution could be to do morphological analysis, which is out of the scope of this work and can be done as a future enhancement.

One final observation that needs attention is the ratio of real word errors. 25% of the typing errors in Urdu are real word errors. This emphasizes the need of some sophisticated real word error detection and correction mechanism. Incorporating the real word error correction can be another possible enhancement.

References

Alberga, C.N. String Similarity and Misspelling, *In Communications of ACM*, Vol. 10, No. 5, pp. 302-313, May, 1967.

Amir, A et. al., Indexing and dictionary matching with one error. (Extended Abstract)

Appel, A. and Jacobson, G. 1988. The world's fastest scrabble program. *Communications of the ACM* 31, 5 (May), 572–578.

Bagwell, 2002. Ideal Hash Trie.

Bagwell, P. 2000. Fast and space efficient trie searches. *Technical Report, EPFL*

Bentley, J. and Sedgwick, R. 1997. Fast algorithms for sorting and searching strings. *In Eighth Annual ACM-SIAM Symposium on Discrete Algorithms* (1997), SIAM Press (1997).

Bobrow, D. G. "Computational linguistics" In *Communications of ACM*, Volume 10(3), pages 302-313.

Brill, E. and Moore, R. C. An Improved Error Model for Noisy Channel Spelling Correction. *In proceedings of 38th Annual meeting of Association for Computational Linguistics*, pp. 286-293, 2000.

Brodal, G.S., Approximate Dictionary Queries ,BRICS*** Computer Science Department

Christian, Peter, Soundex – can it be improved?, *Computers in Genealogy* Vol. 6, No. 5, March, 1998.

Damerau, F.J. A Technique for Computer Detection and Correction of Spelling Errors. *In Communications of ACM*, Vol. 7, No. 3, pp. 171-177, March, 1964.

Erikson, K. Approximate Swedish Name Matching - Survey And Test Of Different Algorithms, 1997.

Fisher, M.W. A statistical text to phone function using ngrams and rules. *In Proceedings Of The IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 649-662, 1999.

Fredkin, E. 1960. Trie memory. *Communications of the ACM* 3, 490–499.

Golding, A. R. A Bayesian Hybrid Method for Context Sensitive Spelling Correction, May 1995.

Holmes, David and McCabe, M. C., Improving precision and recall for Soundex

Hodge, V. J. & Austin, J., A Comparison of Standard Spell Checking Algorithms and a Novel Binary Neural Approach. *IEEE transactions on knowledge and data engineering*, Vol. 15, No. 5, September 2003.

Jan Daciuk and Gertjan van Noord, Finite Automata for Compact Representation of Tuple Dictionaries, Elsevier Science, 17 January 2003.

Jurafsky, D. and Martin, J. H., Speech and Language Processing.: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition Prentice Hall; 1st edition (January 26, 2000)

Kann, Viggo et. al., Implementation Aspects And Applications Of A Spelling Correction Algorithm, May 1998.

Kernighan et. al. A Spelling Correction Program Based on Noisy Channel Model, In Proceedings of COLING-90, *The 13th International Conference On Computational Linguistics*, Vol 2. 1990.

Khan, Rasheed Hasan, "Urdu Imla", Qaumi Council bra-e-Taraki-e-Urdu Zabaan, 1998.

Kukich, K. Techniques for automatically correcting words in text, *ACM Computing Survey*, Vol. 14, No. 4, pp 377-439, December 1992.

Marcin G. Ciura, Sebastian Deorowicz, How to squeeze a lexicon, Practice and Experience 2001; 31(11):1077–1090.

Neagle, R. Soundexfaces - Natural Language Processing (AR32), November 2000.

Peterson, L.J. A Note on Undetected Typing Errors. *In Communications of ACM*, Vol. 29, No. 7, pp. 633-637, July, 1986.

Stanier, Alan, How accurate is Soundex matching, *Computers in Genealogy* Vol. 3, No. 7, pp. 286-288. September 1990.

Toutanova, K. and Moore, R. C. Pronunciation Modeling for Improved Spelling Correction, *In proceedings of 40th Annual meeting of Association for Computational Linguistics*, July 2002, pp. 144-151.

Turba, T. N. 1981. Checking for spelling and typographical errors in computer based text. SIGPLANSIGOA Newslett. (June), 51-60.

Tong, X. et. al. , A Statistical Approach To Automated OCR Error Correction In Context.

Udi Manber and Sun Wu. An Algorithm for Approximate Membership Checking With Application to Password Security, *Information Processing Letters* 50 (1994) 191-197

Witten, Ian. H., Storing And Retrieving Keys In A Table By Cross-Indexing

Zobel, J., Et. Al. Finding Approximate Matches In Large Lexicons, October 1994.

Zobel, J., Et. Al. Searching Large Lexicon For Partially Specified Terms.

Zobel, J. and Dart, P. Phonetic String Matching: Lessons from Information Retrieval.

Appendices

Appendix A. Levenshtien Edit Distance Algorithm

Step	Description
1	Set n to be the length of s. Set m to be the length of t. If n = 0, return m and exit. If m = 0, return n and exit. Construct a matrix containing 0..m rows and 0..n columns.
2	Initialize the first row to 0..n. Initialize the first column to 0..m.
3	Examine each character of s (i from 1 to n).
4	Examine each character of t (j from 1 to m).
5	If s[i] equals t[j], the cost is 0. If s[i] doesn't equal t[j], the cost is 1.
6	Set cell d[i,j] of the matrix equal to the minimum of: a. The cell immediately above plus 1: d[i-1,j] + 1. b. The cell immediately to the left plus 1: d[i,j-1] + 1. c. The cell diagonally above and to the left plus the cost: d[i-1,j-1] + cost.
7	After the iteration steps (3, 4, 5, 6) are complete, the distance is found in cell d[n,m].

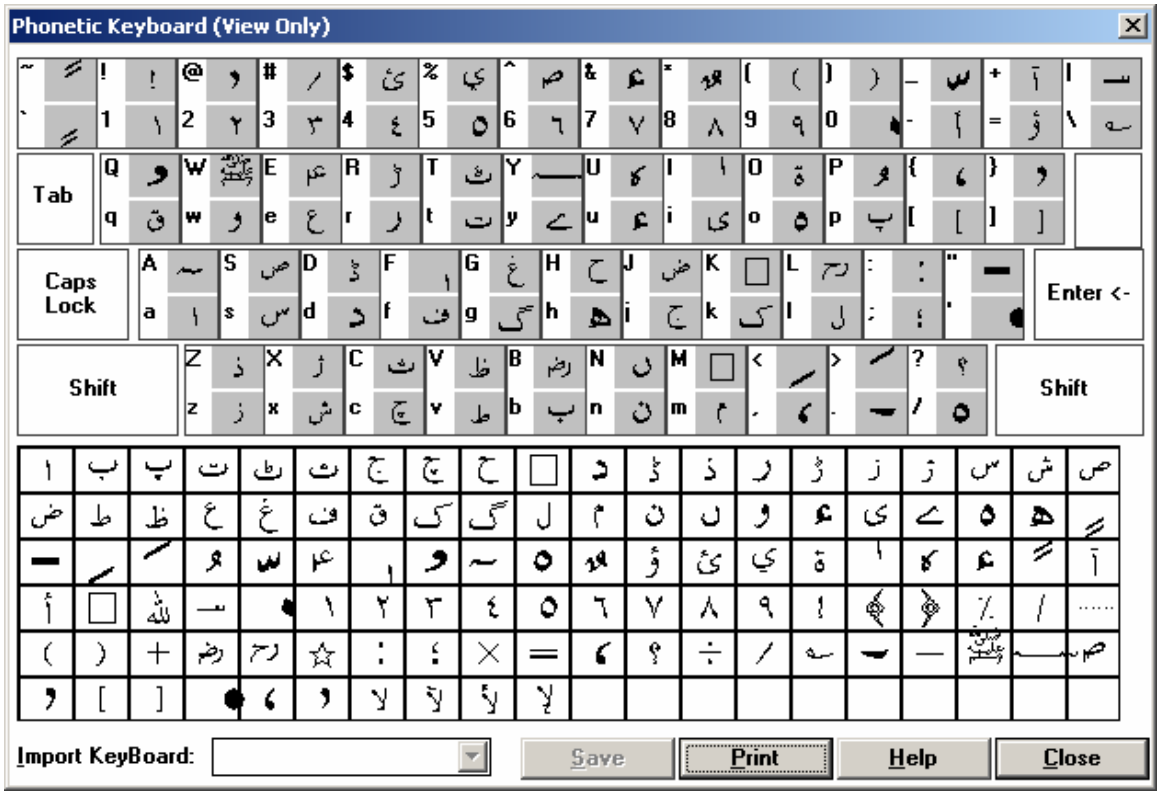
Appendix B. Keyboard Layouts

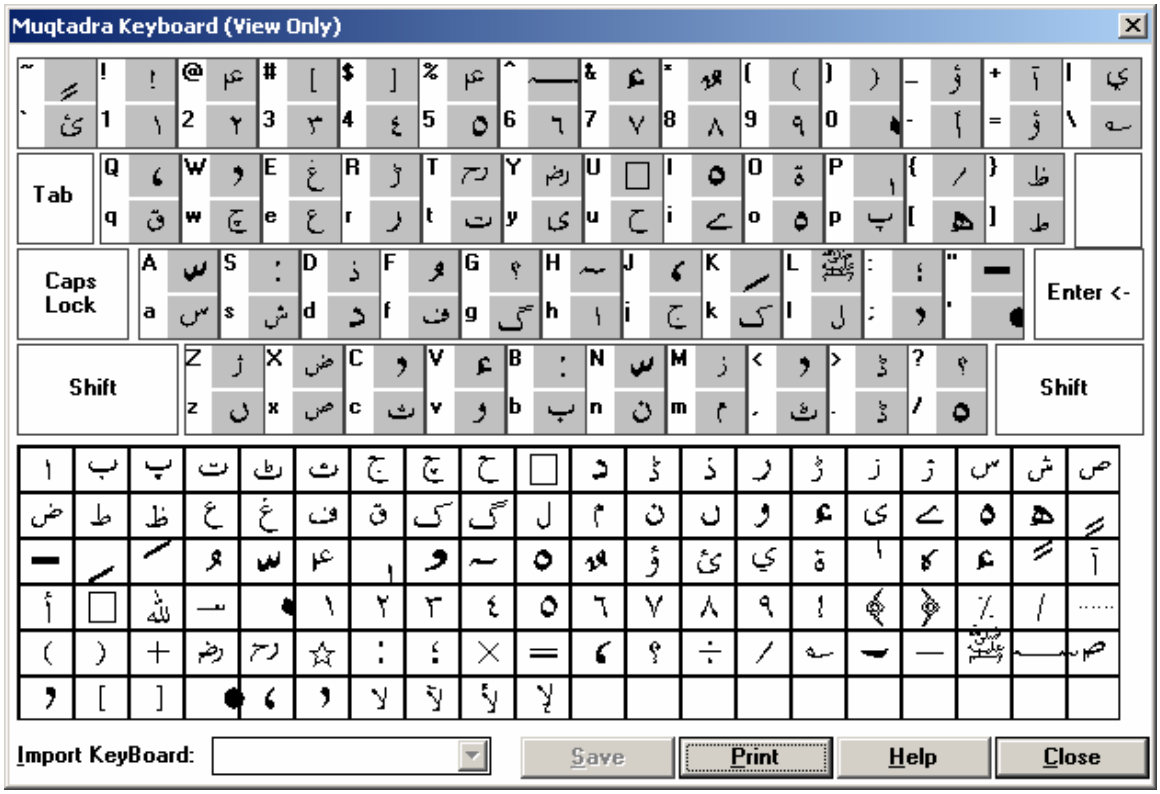
Microsoft Keyboard Layout

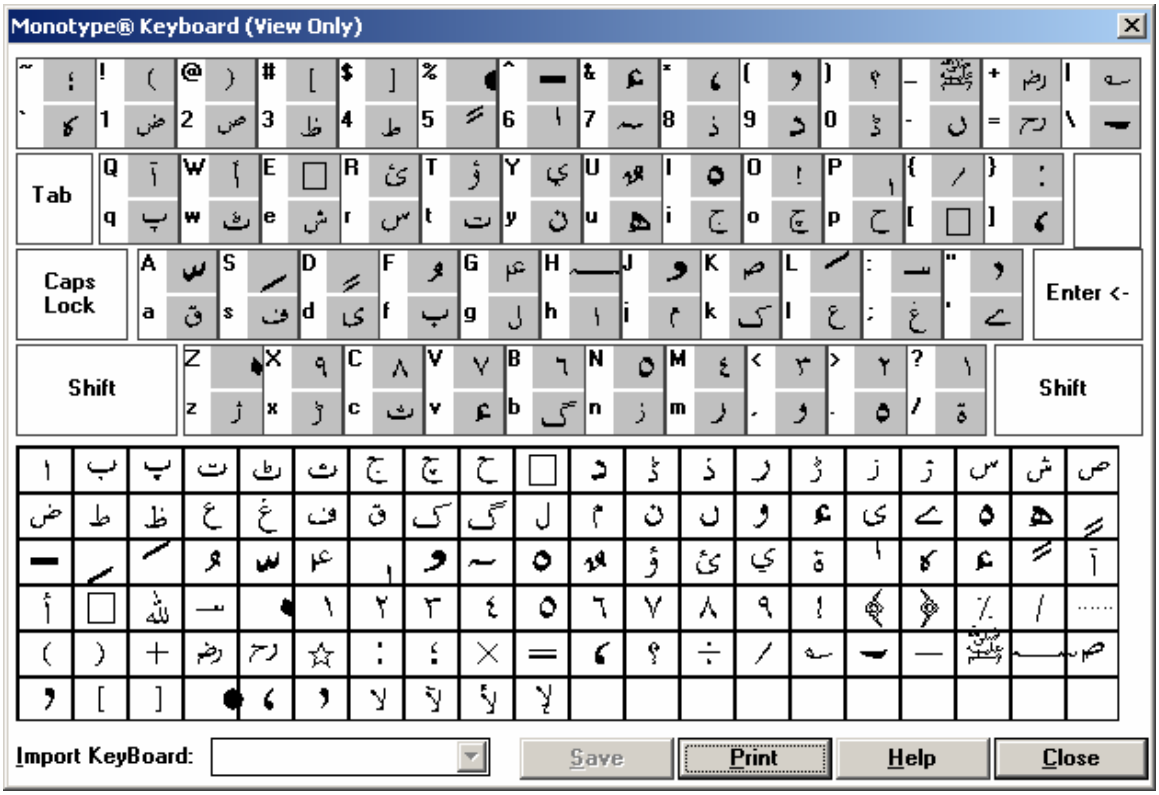
Normal:

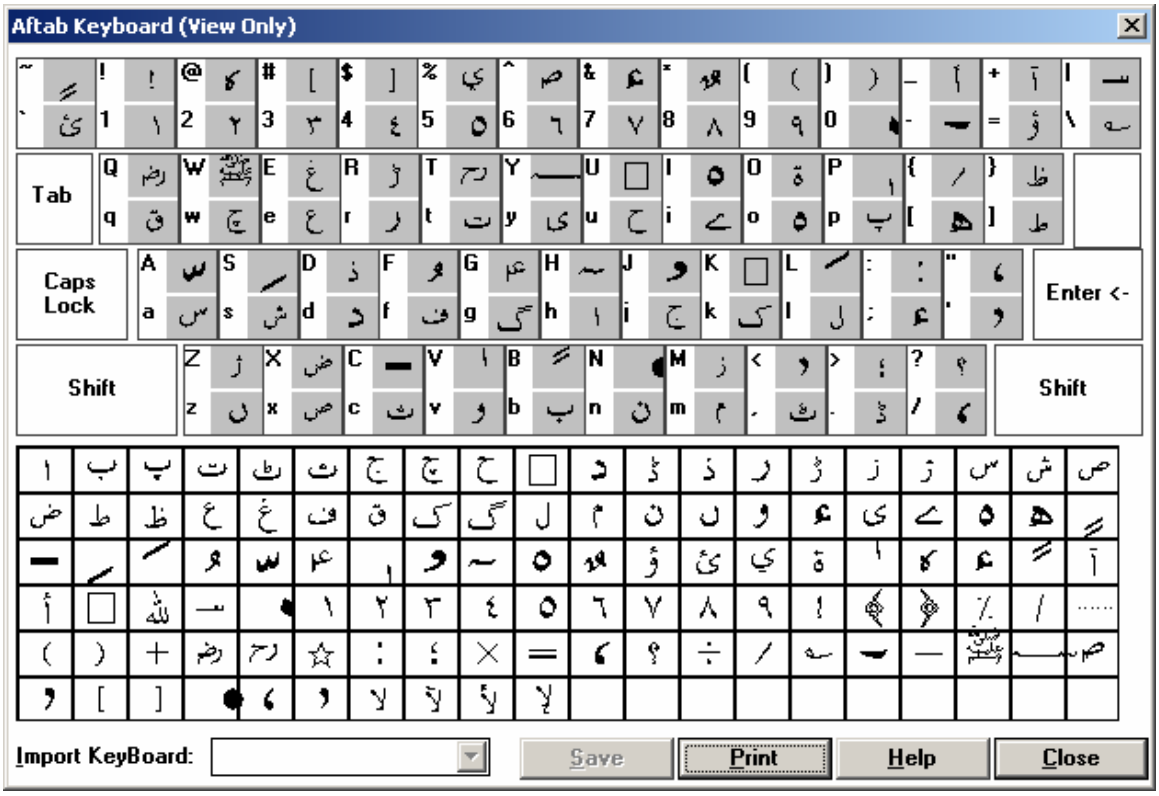
With shift key pressed:

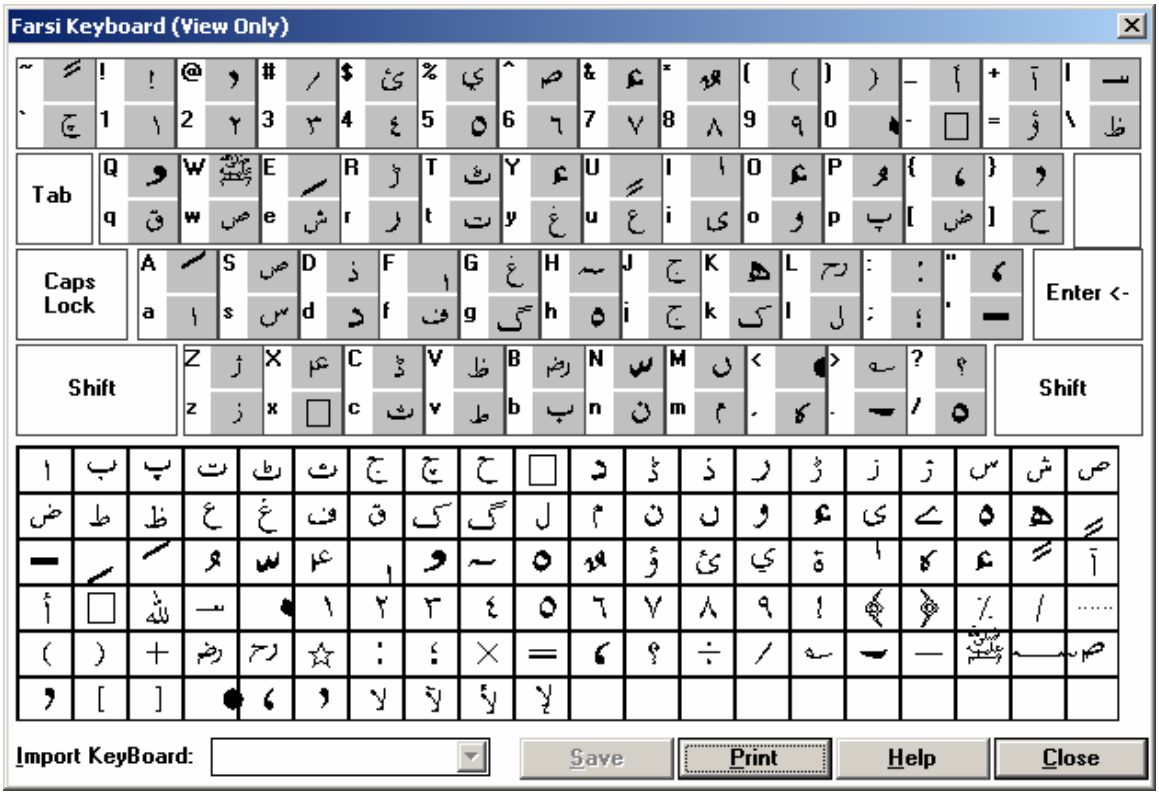
Inpage Keyboard Layouts






Monotype® Keyboard (View Only)
Tab
Caps Lock
Enter <-
Shift
Shift
Import KeyBoard:
Save
Print
Help
Close


Aftab Keyboard (View Only)
Tab
Caps Lock
Enter <-
Shift
Shift
Import KeyBoard:
Save
Print
Help
Close


Farsi Keyboard (View Only)
Tab
Caps Lock
Enter <-
Shift
Shift
Import KeyBoard:
Save
Print
Help
Close

Appendix C. Space Deletion Errors Correction Algorithm

```
NJposesList : list of non joiner characters positions in the Error.
Error : String

start:=0, end:=Error.length-1, startindex := 0 , endindex := size of NJposesList

Function GetParts (start, end, startindex, endindex)

BigCorList : List of partitions of a word
CorPartList : List of parts of a partition
subword : String

for index:= startindex to endindex
                subword := Error.Substring(start,NJposes[index] - start+ 1)
                frequency := isValidWord(subword)
                If frequency != -1 Then
                                cor = New correction()
                                cor.word = subword
                                cor.Frequency = frequency
                   CorPartList := GetParts(NJposes[index] + 1, Error.Length - 1, index +
1, endindex)
                   If CorPartList.Size>0 Then
                      CorPartList = Merge(cor, CorPartList)
                      BigCorList = AppendList(BigCorList, CorList)
                                End If
                End If
End for

return BigCorList

End Function

Function Merge(Cor, CorPartList)

                For index = 1 To CorPartList.size
                                CorPartList[i].word = Cor.word + " " + CorPartList[i].word

                                If Cor.Frequency < CorPartList[i].Frequency1 Then
                                                CorPartList[i].Frequency = Cor.Frequency
                                                //the least frequent part tells the frequency of
whole partition

                                End If
                End For
```

```
                return CorPartList

End Function

Function AppendList(list1, list2)

                For index := 1 To list2.size
                                list1.Add(list2[i])
                End For

                return list1

End Function
```

Appendix D. List of Errors

Errors	Corrects
اکیتس	اکتیس
آروز	آرزو
اخیتار	اختیار
ادراہ	ادارہ
ادورا	ادوار
انساینت	انسانیت
انپی	اپنی
انیتس	انتیس
آرزدہ	آزردہ
آفرنیش	آفرینش
اکھٹا	اکٹھا
بتدیلی	تبدیلی
برادشت	برداشت
ہیئت	ہئیت
محتنانہ	محنتانہ
اچھتلے	اچھلتے
طبعیت	طبیعت
ذیابطیس	ذیابیطس
جادواں	جاوداں
بیسار	بسیار
معمالات	معاملات
حمال	حامل
کپیٹن	کیپٹن
مںی	میں
اکایئوں	اکائیوں
گیئ	گئی

ہیی	یہی
Errors	Corrections
آزمایش	آزمائش
اشیرباد	آشیرباد
انگشتانا	انگشتانہ
اھمیت	اہمیت
بازووں	بازوؤں
آذاد	آزاد
آذادی	آزادی
آذان	اذان
آرذو	آرزو
اجلت	عجلت
ادہر	ادھر
اساتدہ	اساتذہ
افوال	افعال
انتحاب	انتخاب
انکھوں	آنکھوں
انگریری	انگریزی
انگیر	انگیز
انہراف	انحراف
انہصار	انحصار
اژدھام	اژدہام
اژوہام	اژدہام
اکانوئے	اکانوے
اہستہ	آہستہ
ایصا	ایضا
بادباں	بادبان

بارھا	بارہا
باعباں	باغباں
Errors	Corrections
باذ	باز
بانب	جانب
اہلیا	اہلیہ
اعراز	اعراض
انشان	انسان
اراہ	ارادہ
ارھر	ادھر
آدا	ادا
انکساف	انکشاف
بدیہ	ہدیہ
برانگیخنگی	برانگیختگی
برگز	ہرگز
بزداں	یزداں
موثر	مؤثر
کھیٹ	کھیت
نمونا	نمونہ
سرمایے	سرمائے
جاءزہ	جائزہ
آسایش	آسائش
بدیشی	بدیسی
زایل	زائل
کنٹاپ	کنٹوپ
اکادت	اکارت
تمہیں	تمھیں
لکتر	بکتر

تشبیہائی	تشبیہاتی
ذرج	درج
طلاطم	تلاطم
Errors	Corrections
زبائی	زبانی
کنبد	گنبد
قضہ	قصہ
معنک	معنی
شائد	شاید
کئے	کیے
ذانی	ذاتی
مونت	مونث
فرشتا	فرشتہ
بالاخر	بالآخر
مدس	مدت
منالک	منسلک
ززاق	رزاق
محفوط	محفوظ
قایم	قائم
پزیر	پذیر
مرتیہ	مرتبہ
پچھا	بچھا
طرخ	طرح
ھا	جا
جگاوری	جگادری
مرست	مرمت
چسپاد	چسپاں
سکر	سکڑ

ذرد	زرد	لحاض	لحاظ
ذیب	زیب	وجور	وجود
ذینت	زینت	رورانیے	دورانیے
ترکی	ترقی	ذبان	زبان
Errors	Corrections	وچود	وجود
بینالاقاومی	بینالاقوامی	Errors	Corrections
دل	دس	حزف	حذف
شایع	شائع	ملاحضه	ملاحظه
مطمین	مطمئن	اودو	اردو
ہربه	حربه	زمه	ذمه
زروری	ضروری	مین	میں
بزله	بذله	اگله	اگلا
رزک	رسک	پھون	بھون
زبددست	زبردست	مودوں	مردوں
داماندگی	واماندگی	بیچھے	پیچھے
نفاظ	نفاذ	چاہئے	چاہیے
فروع	فروغ	سیاخوں	سیاحوں
پزیری	پذیری	مداخوں	مداحوں
آرایش	آرائش	مباچه	مباحثه
حوارث	حوادث	اکر	اگر
ابنگلو	اینگلو	بنرش	بندش
اعتیار	اعتبار	روباره	دوباره
فود	فوڈ	پایپ	پائپ
موازته	موازنه	ررخت	درخت
گذیر	گزیر	اقص	عکس
سحزی	سحری	عائضری	عارضی
خصار	حصار	کیا	کئی
زریعے	ذریعے	ینہں	نہیں

طافوں	لحافوں
عنقاد	انعقاد
تے	جاتے
نارص	ناراض
دستبرد	دسترس
اشعاروں	اشعار
Errors	Corrections
آزاردگی	آزردگی
اداراک	ادراک
ارتقا ع	ارتقا
المشیرقین	المشرقین
امیداواروں	امیدواروں
امیداواری	امیدواری
انتاللیس	انتالیس
انختیار	اختیار
اندازاہ	اندازہ
آتشنردگی	آتشزدگی
اٹھکھیلیاں	اٹکھیلیاں
اپنبے	اپنے
استشناء	استثنا
انتفاء	انتفا
انکڑنا	اکڑنا
ایکلوتا	اکلوتا
استخباراری	استخباری
ئکلیات	کلیات
ؤگھات	گھات
اارتقا	ارتقا
گندلا	گدلا

ٴمیں	میں
فاؤسٹ	فاسٹ
ٹہنییوں	ٹہنیوں
بڑابڑاہٹ	بڑبڑاہٹ
آڑھے	آڑے
افکارات	افکار
افساننے	افسانے
Errors	Corrections
عیسویں	عیسوی
خمکار	خمار
معلوما	معلوم
چاہیئے	چاہیے
چاہےئے	چاہیے
مئوقف	مؤقف
میںض	میں
اآصف	آصف
ساتھض	ساتھ
ماہین	ماہرین
کمہ	کہ
مومنث	مونث
ہومارا	ہمارا
ٹھوکریاں	ٹوکریاں
شادباش	شاباش
بدمزدگی	بدمزگی
دیتں	دیں
بظاہرا	بظاہر
ضبطر	ضبط
دیکھاےئے	دکھاےئے

سوہیل	سہیل
تائثر	تاثر
برایں	بریں
یپہاں	یہاں
گھو	گو
سلالم	سلام
ہاگں	ہاں
ہہوتا	ہوتا
Errors	Corrections
مصنوروں	مصوروں
کاختتام	اختتام
آرزؤں	آرزوؤں
آزادنه	آزادانه
آزدانه	آزادانه
ارده	اراده
اعترض	اعتراض
اعترضات	اعتراضات
اقتدر	اقتدار
انتخبات	انتخابات
انداج	اندراج
انفردی	انفرادی
انگلیڈ	انگلینڈ
ایشائی	ایشیائی
باوجو	باوجود
بتریج	بتدریج
بذریه	بذریعه
اولولعزمی	اولوالعزمی
اچنبے	اچنبھے

آدات	عادات
بانات	بیانات
ئنهٔ	آئنهٔ
امانده	وامانده
سنکرت	سنسکرت
دخت	دختر
آستنیں	آستینیں
افزائ	افزائی
روپه	روپیه
Errors	Corrections
جھملاتا	جھلملاتا
کنڑول	کنٹرول
سینئر	سینیئر
انجینئر	انجنیئر
نفسات	نفسیات
پامال	پائمال
لیکشن	الیکشن
مزاحت	مزاحمت
ٹہرا	ٹھہرا
اسملی	اسمبلی
مصد	مصدر
پھلجڑی	پھلجھڑی
دکاندار	دوکاندار
طریوں	طریقوں
ور	اور
ازں	ازاں
جنیاتی	جینیاتی
زول	زوال

جھائے	جھکائے
جھے	مجھے
ک	کو
تشخصی	تشخیصی
فراد	افراد
شمار	مشمار
ہنی	ہونی
علاق	علاقے
حثیت	حیثیت
ہفے	ہفتے
Errors	Corrections
ت	تو
متخب	منتخب
مذاکات	مذاکرات
مک	ملک
خوہ	خواہ
آزدی	آزادی
دربا	دربار
ہتی	ہوتی
میونسل	میونسپل
متلا	مبتلا
متزاج	امتزاج
نووانوں	نوجوانوں
آنکوں	آنکھوں
را	رہا
پناں	پنہاں
وق	وقت
آسو	آنسو

تجزہ	تجزیہ
لسنیات	لسانیات
آاقی	آفاقی
الاظ	الفاظ
ادراجی	اندراجی
مابق	مطابق
حلی	حلقی
حف	حرف
ذیر	زیر
دلآویزی	دلاویزی
آنہیں	آ نہیں
Errors	Corrections
آبیٹھی	آ بیٹھی
آجا	آ جا
آجائیں	آ جائیں
آجارہے	آ جا رہے
آدھمکا	آ دھمکا
آسکتے	آ سکتے
آسکی	آ سکی
آعندلیب	آ عندلیب
آلیتا	آ لیتا
آپڑنا	آ پڑنا
آپہنچا	آ پہنچا
آپہنچے	آ پہنچے
آچکا	آ چکا
آگری	آ گری
آگھسی	آ گھسی
آکر	آ کر

ونظر	و نظر	وجذبات	و جذبات
وساکن	و ساکن	آگئے	آ گئے
آرہاتھا	آ رہا تھا	آگئی	آ گئی
وپنچ	و پنچ	وسکنات	و سکنات
آجاتے	آ جاتے	وفکر	و فکر
آرہی	آ رہی	ونازک	و نازک
آجاتی	آ جاتی	واسلوبیاتی	و اسلوبیاتی
آجانا	آ جانا	ومعروف	و معروف
آجاتا	آ جاتا	ونزار	و نزار
وسباق	و سباق	واستعاراتی	و استعاراتی
ومفہوم	و مفہوم	ومزاح	و مزاح
آلگا	آ لگا	وتمدن	و تمدن
Errors	Corrections	وغریب	و غریب
آجانے	آ جانے	Errors	Corrections
آرہے	آ رہے	وجوار	و جوار
آرہا	آ رہا	ونمود	و نمود
وجواب	و جواب	وبھارت	و بھارت
آموجود	آ موجود	اسلیے	اس لیے
وشور	و شور	برآنا	بر آنا
وخصائل	و خصائل	برآنے	بر آنے
وغایت	و غایت	آؤگی	آؤ گی
واحساسات	و احساسات	ازسرنوروشن	از سر نو روشن
آجاتیں	آ جاتیں	اسمیں	اس میں
وخیالات	و خیالات	انبچوں	ان بچوں
وکرم	و کرم	برمساوات	بر مساوات
وملال	و ملال	پرامید	پر امید
وواقعات	و واقعات	بدنظمی	بد نظمی
ومشقت	و مشقت	ترتھا	تر تھا

کواچھوتا	کو اچھوتا
بیخود	بے خود
جوکه	جو که
سرانجام	سر انجام
یکدیگر	یک دیگر
آردار	آر دار
درفشانی	در فشانی
سرگرانی	سر گرانی
توکرکٹ	تو کرکٹ
جاکر	جا کر
کاکردار	کا کردار
دیکر	دے کر
بیباکی	بے باکی
جوافسانے	جو افسانے
Errors	Corrections
ہوجاتا	ہو جاتا
ہوتو	ہو تو
کرلیں	کر لیں
سرچکرا	سر چکرا
کابیان	کا بیان
سرہلنے	سر ہلنے
کررکھا	کر رکھا
بہوش	بے ہوش
کوباکسنگ	کو باکسنگ
سرمنڈھ	سر منڈھ
مدوجزر	مد و جزر
ہوگئے	ہو گئے
دلعزیز	دل عزیز

سرتن	سر تن
غمزده	غم زده
بامعنی	با معنی
درپرده	در پرده
ہوگیا	ہو گیا
لاکر	لا کر
پرفائز	پر فائز
پرمنکشف	پر منکشف
ہواور	ہو اور
کوجھنجھوڑ	کو جھنجھوڑ
کاعطیه	کا عطیه
جانہیں	جا نہیں
تووه	تو وه
یاشاید	یا شاید
پراثر	پر اثر
Errors	Corrections
کانقطه	کا نقطه
کرلیتا	کر لیتا
کوکئی	کو کئی
ہوسکتا	ہو سکتا
ہوچکے	ہو چکے
کوصرف	کو صرف
ازحد	از حد
باضابطه	با ضابطه
برسراقتدار	بر سر اقتدار
بروقت	بر وقت
پرآشوب	پر آشوب
پرپورے	پر پورے

Errors	Corrections
پرمہربانیاں	پر مہربانیاں
جاسکتا	جا سکتا
جاسکتی	جا سکتی
جاسکیں	جا سکیں
جاسکے	جا سکے
دونکات	دو نکات
ردعمل	رد عمل
سرفہرست	سر فہرست
مدنظر	مد نظر
نوتشکیل	نو تشکیل
کاآسان	کا آسان
کاتعین	کا تعین
کاحلف	کا حلف
کاخواب	کا خواب
کرجاتا	کر جاتا
کردو	کر دو
کردوں	کر دوں
کردیا	کر دیا
کردیتا	کر دیتا
کردیتے	کر دیتے
کردے	کر دے
کرسکتا	کر سکتا
کرسکتے	کر سکتے
کرسکی	کر سکی
کرسکیں	کر سکیں
کرسکے	کر سکے
کرلیا	کر لیا

Errors	Corrections
کرکے	کر کے
کووحشت	کو وحشت
ہرشعبے	ہر شعبے
ہوچکا	ہو چکا
ہوسکا	ہو سکا
ہوسکتی	ہو سکتی
ہوسکتے	ہو سکتے
ہوسکی	ہو سکی
ہوگئی	ہو گئی
ہوگا	ہو گا
ہوگی	ہو گی
بیحرمتی	بے حرمتی
پرمشرف	پر مشرف
جاچکا	جا چکا
جارہے	جا رہے
ملکر	مل کر
کااظہار	کا اظہار
Errors	Corrections
کاقبضہ	کا قبضہ
کردی	کر دی
کررہی	کر رہی
کررہے	کر رہے
کرسکتیں	کر سکتیں
کواگلے	کو اگلے
کوگھروں	کو گھروں
کوکچھ	کو کچھ
کوہم	کو ہم
ہورہی	ہو رہی

ہورہے	ہو رہے
ہوسکے	ہو سکے
کردیئے	کر دیئے
کرچوتھے	کر چوتھے
مجکو	مج کو
امروارادے	امر و ارادے
امروجودی	امر وجودی
اورارادے	اور ارادے
اباواجداد	ابا و اجداد
میںفوری	میں فوری
اورادنی	اور ادنی
آؤنگا	آؤں گا
آؤنگی	آؤں گی
آیاتک	آیا تک
آیاتھا	آیا تھا
آیاوہ	آیا وہ
اتررہا	اتر رہا
اترچکا	اتر چکا
Errors	Corrections
اثراقبال	اثر اقبال
اثربہت	اثر بہت
اثرہے	اثر ہے
ادادل	ادا دل
اداکرتی	ادا کرتی
اردگردکی	ارد گرد کی
اسےصبح	اسے صبح
امرلامتناہی	امر لا متناہی
امرمترشح	امر مترشح

بتادینا	بتا دینا
بتادے	بتا دے
بتاسکتا	بتا سکتا
بتاسکتی	بتا سکتی
پریرو	پری رو
چندہم	چند ہم
باردگر	بار دگر
اورجگہ	اور جگہ
جانثار	جان نثار
اورجو	اور جو
ذراسی	ذرا سی
اوررنگ	اور رنگ
تنگدل	تنگ دل
شبوکی	شبو کی
تھاان	تھا ان
اناسے	انا سے
گیاہے	گیا ہے
عودکر	عود کر
اوراس	اور اس
Errors	Corrections
خوددار	خود دار
برفباری	برف باری
اورلڑکی	اور لڑکی
اورتقدیر	اور تقدیر
بھردیتے	بھر دیتے
ہواہے	ہوا ہے
چنددنوں	چند دنوں
اورراوی	اور راوی

پھربتدریج	پھر بتدریج
پھرمدعو	پھر مدعو
اورکافی	اور کافی
اورپورا	اور پورا
بنادیتا	بنا دیتا
بناسکیں	بنا سکیں
بڑاعمل	بڑا عمل
بہرطور	بہر طور
پارکریں	پار کریں
تھااور	تھا اور
شکرہے	شکر ہے
غیرسرکاری	غیر سرکاری
غیرسیاسی	غیر سیاسی
غیرمسلح	غیر مسلح
غیرمعمولی	غیر معمولی
فخرمحسوس	فخر محسوس
نظرنہیں	نظر نہیں
نیزاب	نیز اب
کیاتھا	کیا تھا
ہواجو	ہوا جو
Errors	Corrections
اورپارلیمنٹ	اور پارلیمنٹ
بناسکتے	بنا سکتے
پڑاگورنر	پڑا گورنر
تھاکہ	تھا کہ
چلالیا	چلا لیا
زوردیا	زور دیا
میرواعظ	میر واعظ

کریگا	کرے گا
کہاکہ	کہا کہ
کہاہے	کہا ہے
ہونگے	ہوں گے
اورآلو	اور آلو
کیاکہنا	کیا کہنا
اسیرنظر	اسیر نظر
افزانشانیاں	افزا نشانیاں
پیلانشان	پیلا نشان
آئیں‌گی	آئیں گی
آزادکرانے	آزاد کرانے
آزادہو	آزاد ہو
اتارلیں	اتار لیں
اتناقریب	اتنا قریب
اجزاکو	اجزا کو
احمدبن	احمد بن
احمدصاحب	احمد صاحب
اخترعشرت	اختر عشرت
اردوصحافت	اردو صحافت
اردومیں	اردو میں
اردونہیں	اردو نہیں
Errors	Corrections
ازاں‌اس	ازاں اس
ازہرکے	ازہر کے
ازیں‌ظہور	ازیں ظہور
اشیاتک	اشیا تک
اعظمکی	اعظم کی
امورہمارے	امور ہمارے

امیدکا	امید کا
اندرتاثرات	اندر تاثرات
اندرتھا	اندر تھا
اندرجھانکا	اندر جھانکا
اندرحضور	اندر حضور
اندرسمو	اندر سمو
بالابحث	بالا بحث
بالاسلسلہ	بالا سلسلہ
باہرجانے	باہر جانے
باہرنکلنے	باہر نکلنے
باہرکیا	باہر کیا
باہرگھمانے	باہر گھمانے
برترحقیقت	برتر حقیقت
برترکا	برتر کا
برپاکردیا	برپا کر دیا
کملابائی	کملا بائی
وزیراعظم	وزیر اعظم
اتراہوا	اترا ہوا
کرونگی	کروں گی
مصدرہٹا	مصدر ہٹا
سکتاغائر	سکتا غائر
بھیاکا	بھیا کا
Errors	Corrections
قطاریوں	قطار یوں
لگیں‌آ	لگیں آ
موٹرسائیکل	موٹر سائیکل
دیناہے	دینا ہے
انورکی	انور کی

جاتاہے	جاتا ہے
ملتاہے	ملتا ہے
چاندکے	چاند کے
بغیروہ	بغیر وہ
گینددیکھی	گیند دیکھی
اکثرشائع	اکثر شائع
آوازکے	آواز کے
پانیوالے	پانے والے
جائیگا	جائے گا
جہاندیدہ	جہاں دیدہ
لگورہا	لگوا رہا
نوکرشاہی	نوکر شاہی
پیدانہیں	پیدا نہیں
پیداہواتو	پیدا ہوا تو
کرنیوالے	کرنے والے
کرینگے	کریں گے
ہزارڈالر	ہزار ڈالر
اندازحکومت	انداز حکومت
ابتدااب	ابتدا اب
ابتداہی	ابتدا ہی
ابھاراور	ابھار اور
اتروالیتیں	اتروا لیتیں
اجاڑاہے	اجاڑا ہے
Errors	Corrections
احسانلاہور	احسان لاہور
اخباروطن	اخبار وطن
اخبارکے	اخبار کے
ارتقاکا	ارتقا کا

ارتقاکے	ارتقا کے
ارسطوکے	ارسطو کے
اسراراب	اسرار اب
اصراربڑھا	اصرار بڑھا
اصرارتھا	اصرار تھا
اصرارنے	اصرار نے
اطوارکی	اطوار کی
اظہارذاتی	اظہار ذاتی
اظہارکرتا	اظہار کرتا
اعجازدہلوی	اعجاز دہلوی
افرادمیں	افراد میں
افرادکی	افراد کی
افکارکی	افکار کی
افکارہی	افکار ہی
اقبالاپنی	اقبال اپنی
اقداراور	اقدار اور
امدادکے	امداد کے
اندازسے	انداز سے
اندازنہ	انداز نہ
اندازکر	انداز کر
اندازکرنے	انداز کرنے
اندازہوں	انداز ہوں
انساںآخری	انساں آخری
انساںاپنی	انساں اپنی
Errors	Corrections
انواراقبال	انوار اقبال
انکارکا	انکار کا
انہیںبہت	انہیں بہت
انہیںخاص	انہیں خاص
انہیںکہیں	انہیں کہیں
بالوںکے	بالوں کے
بتایاتھا	بتایا تھا
بتیاںہیں	بتیاں ہیں
ڈھلانپر	ڈھلان پر
سنہراپلو	سنہرا پلو
سجاناچاہتا	سجانا چاہتا
سمجھالیکن	سمجھا لیکن
کردارہیں	کردار ہیں
مناظرکی	مناظر کی
بتاتاہے	بتاتا ہے
اظہارخیال	اظہار خیال
بنیادقرار	بنیاد قرار
دیانتدار	دیانت دار
گفتگوان	گفتگو ان
ہوئنگے	ہوئیں گے
بکھراہونے	بکھرا ہونے
انتشارواافتراق	انتشار و افتراق
برصغیرکے	برصغیر کے
آدمیوںکو	آدمیوں کو
آویزاںہوا	آویزاں ہوا
اجتہادسے	اجتہاد سے
احترازکرتے	احتراز کرتے
اختیارمیں	اختیار میں
Errors	Corrections
اختیارکرتے	اختیار کرتے
اختیارکرلیتی	اختیار کر لیتی

Errors	Corrections
اختیارکیا	اختیار کیا
اختیارکے	اختیار کے
اداروںکے	اداروں کے
ارمغانمیں	ارمغان میں
استوارکریں	استوار کریں
اعتمادکرنے	اعتماد کرنے
افسروںاور	افسروں اور
اقتداراور	اقتدار اور
امتیازہی	امتیاز ہی
امرتسروالی	امرتسر والی
امنگوںکا	امنگوں کا
انتظارکرنے	انتظار کرنے
انصاریبھی	انصاری بھی
باوجودبھی	باوجود بھی
باچھیںکھل	باچھیں کھل
برصغیرکی	برصغیر کی
برقراربھی	برقرار بھی
برقاررکھوگے	برقرار رکھو گے
برقاررہنا	برقرار رہنا
جدوجہدکی	جدوجہد کی
اہلکارنکالے	اہلکار نکالے
استبدادنه	استبداد نه
استبدادکی	استبداد کی
استعدادکی	استعداد کی
انسانوںکو	انسانوں کو
انسانوںکی	انسانوں کی
انٹرویوکر	انٹرویو کر
انگلینڈکے	انگلینڈ کے
انیسویںصدی	انیسویں صدی
اندھیراہوگا	اندھیرا ہو گا
انقلابوںکے	انقلابوں کے
برساتیاںبھی	برساتیاں بھی
جاؤنگا	جاؤں گا
اوراردو	اور اردو
سبکے	سب کے
حدتکآپ	حد تک آپ
ہیکه	ہے که
بھیلکھا	بھی لکھا
تاجمحل	تاج محل
رکنکے	رکن کے
روینگے	رہیں گے
ہوئیاور	ہوئی اور
اپنیمرضی	اپنی مرضی
حالتمیں	حالت میں
ادارہپورے	ادارہ پورے
حکومتکے	حکومت کے
دیکھیجا	دیکھی جا
سلامتیکے	سلامتی کے