

Isolated-word Error Correction for Partially Phonemic Languages using Phonetic Cues

**Bhupesh Bansal, Monojit Choudhury, Pradipta Ranjan Ray
Sudeshna Sarkar and Anupam Basu**

Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur
West Bengal INDIA 721302

Email: {*bhupesh, monojit, pradipta, sudeshna, anupam*}@cse.iitkgp.ernet.in

Abstract

Partially phonemic languages use writing systems which are in between strictly phonemic and non-phonemic orthography. Therefore, phonetic errors are very frequent in such languages. This paper introduces an approach for development of spellcheckers for partially phonemic languages that use *grapheme-to-phoneme mapping* for isolated-word error correction. Since, a complete and accurate grapheme-to-phoneme system is overkill for a spellchecker, the framework can deal with incomplete phonological information through the use of metaphonemes. The paper also discusses the implementation of a Bengali spellchecker based on this approach and some other issues specific to the Bengali spell-checking. The framework described here is generic in nature and can be used for any partially phonemic languages by incorporating the language specific parts like phonological rules, the keyboard layout and ranking strategies. This approach is very useful for Indian languages as most of them are partially phonemic in nature.

1 Introduction

Spell checking applications or spellcheckers are supposed to detect wrongly typed words and suggest corrections. There are several efficient algorithms for detection of nonword errors. However, prediction of the correct spelling is quite difficult because apart from the typographic error patterns, the keyboard layout and the context, spelling errors in a language are also dependent on the morpho-phonemic structure of the language and the orthographic rules. Most of the commercially available spellcheckers suggest correct spellings taking clues from human typographic error patterns and keyboard layouts, and rank the suggestions either by alphabetical sorting or by considering the monogram and occasionally the bigram frequencies of the suggested word. Nevertheless, the morpho-phonology

or orthographic rules of the language are often not captured in their entirety.

It is a well known fact that phonetic errors are more difficult to correct than other spelling errors as they distort the mis-spelt word more than a single insertion, deletion, substitution or swapping errors [Kukich, 1992]. For example, the Microsoft Word 2002 (10.2627.2625) spellchecker gives no suggestions for the mis-spelt word “eraaknofobia”, which is obviously “arachnophobia” to any human being. Typing “eraaknophobia” or “araknofobia” elicits the correct suggestion from the spellchecker. It seems that Word does not suggest corrections beyond a particular orthographic edit distance, even if the phonetic edit distance is small. Similarly, suggestions for “phether” include “whether”, “hither”, “heather” and “thither” but not “feather”.

We believe that proper modelling of morphophonemic and orthographic rules are key to isolated-word error correction, especially in *phonemic* or *partially phonemic languages* like Hindi, Bengali or Russian. In this paper, we suggest a design of a generic framework involving simple rewrite rules for grapheme to phoneme (G2P) mapping, pattern matching involving *edit distance*, and graph like structure for storing the dictionary of words in both graphemic and phonetic domains. Exceptions to G2P rules can be easily handled by using *metaphonemes*. Such a framework can efficiently predict correct spellings based on phonological cues as well as orthography and typing error patterns. The language-specific G2P rules and information about the keyboard layout can be plugged into the system to generate a spellchecker. A Bengali spellchecker has been implemented using this framework.

The paper is organized as follows. Section 2 introduces the concept of partially phonemic languages and describes their phonological features and spelling error patterns. Section 3 takes a look at some of the existing approaches for developing spellcheckers and elucidates the need of further improvement in case of partially phonemic languages. Section 4 describes the computational framework and related algorithms used for the spellchecker. Section 5 gives a brief outline of a Bengali spellchecker and related implementation issues developed using the framework described in this paper. The concluding section discusses the scope of further improvements and outlines future research.

2 Modelling of Spelling Errors for Partially Phonemic Languages

Phonemic languages use a writing system with a one to one mapping between the letters (grapheme) and the sounds (phoneme). Such writing systems are also referred to as *phonemic orthography* or *true alphabets* [wikipedia, 2004]. Georgian, Esperanto and Sanskrit are examples of strictly phonemic languages. On the other hand languages like English and French use non-phonemic writing systems. This can be illustrated by the English word “station” where the first *t* is pronounced as /t/¹ whereas the second one is pronounced as /sh/. English is a morpho-phonemic language, where the sounds of the morphemes are unambiguous but not that of individual letters.

Partially phonemic languages use writing systems which are in between strictly phonemic and non-phonemic. All *Modern Indo-Aryan languages* (MIA), being derived from Sanskrit, are partially pho-

¹In this paper the graphemes for Indian languages are written in italics using the ITANS notation (www.aczone.com/itrans) and the phonemes are written within two '/' and are in general self explanatory. /s/, /sh/ and /Sh/ represents the alveolar, postalveolar and retroflex unvoiced fricatives. /R/ stands for the retroflex *r*. /E/ stands for *a* in *man* and /e/ stands for *e* in *men*. /O/ stands for *o* in *got*.

mic. They still use Sanskrit orthography although the sounds and pronunciation rules have changed to varying degrees. We take the example of Bengali orthography to illustrate below how it has deviated from the strictly phonemic orthography of Sanskrit[Chatterji, 2002].

- **Phoneme Mergers:** The three phonemes /sh/, /Sh/ and /s/ represented respectively by the graphemes *sh*, *Sh* and *s* of Sanskrit have merged into a single phoneme /sh/ in Bengali only with context dependent allophonic variation although the three graphemes have retained their individual identities. For example the word *shushruShA* is pronounced as /shusrushA/. Here the first *sh* and *Sh* have been mapped to /sh/ and the second *sh* has been mapped to /s/ due to the presence of the following /r/.
- **Single Grapheme for Multiple Phonemes:** Bengali has a single grapheme (*e*) for the vowels /e/ and /E/. Hence, *eka* is pronounced as /Ek/, whereas *megha* is pronounced as /megh/.
- **Vowel Harmony:** Bengali phonology illustrates vowel harmony or more specifically *vowel height assimilation*, which is not reflected in the spellings. For example *patha* is pronounced as /pOth/, whereas *pathika* is pronounced as /pothik/, where due to the following high vowel /i/, /O/ has been raised to /o/.
- **Schwa (a) Deletion:** The examples of *patha* and *pathika* above also illustrates the phenomenon schwa deletion, where the last schwa (*a*) is not pronounced. Bengali also features word-medial schwa deletion (e.g. *bAjanA* /bAjnA/).

There are other phenomena like pronunciations of conjugates or the vowel *RRi* (which is no longer a vowel in modern Bengali but a consonant vowel pair /ri/), which tend to destroy the phonemic nature of Bengali orthography. However, except for some very special cases like schwa deletion and /e/-/E/ distinctions, Bengali G2P is strict and predictable. These factors make us categorize Bengali as a partially-phonemic language. Apart from other MIA like Hindi, Assamese, Marathi etc. languages like Russian, Tamil and Telegu also use partially phonemic-orthography.

Orthographic Word	Pronunciation	Example Error	Graphemic Edit Distance
<i>mumUrShu</i>	/mumurshu/	* <i>mUmurshU</i>	4
<i>chAparAsi</i>	/chAprAshi/	* <i>chAprAshI</i>	3
<i>RRiShi</i>	/rishi/	* <i>rishi</i>	3
<i>tattba</i>	/tOtto/	* <i>tbatya</i>	3

Table 1: Illustrations of some of the common phonemic spelling errors in Bengali. The valid and mis-spelt word pairs have identical pronunciations but quite high orthographic edit distance.

Phonetic errors are far more common in partially phonemic languages than strictly phonemic or non-phonemic languages. In a strictly phonemic language like Sanskrit, the knowledge of proper pronunciation of the word is enough for deducing the correct spelling. On the other hand for non-phonemic languages like English, more effort is spent by the learners to memorize the correct spellings. However, for partially phonemic languages, the knowledge of the correct pronunciation may not be

sufficient to deduce the spelling although it can be quite accurate most of the time. Therefore, speakers tend to overlook the orthography while learning and later commit phonemic errors. Table 1 illustrates some of the common phonemic spelling errors in case of Bengali.

3 Existing Approaches and Possible Improvements

Spell checking applications and related algorithms have been explored since early twentieth century due to the high commercial demand for these applications. [Kukich, 1992] provides a comprehensive survey of spellcheckers, including noword error detection techniques, schemes for isolated-word error correction and context dependent error detection and correction. For space limitations, only the works related to phonetic error detection and correction are discussed in this section with special reference to spellcheckers for Indian languages.

3.1 Phonetic Error Patterns and Correction Schemes

In a study [Van Berkel and DeSmedt, 1988] of spelling errors in Dutch surnames, 38% of the errors were found to be phonetically plausible. For English, Mitton [Mitton, 1987] reported that 44% of all the spelling errors in 925 student essays that he studied were due to homophone substitution. The typographic errors normally change the valid word by only one substitution, deletion, insertion or swapping [Pollock and Zamora, 1984]. Moreover, typographic errors hardly effect the first letter of a word [Pollock and Zamora, 1983; Yannakoudakis and Fawthrop, 1983]. Such assertions are hardly correct for phonetic errors; the cases for both first position error and multiple errors have been illustrated in table 1.

There are different isolated-word error correction techniques based on minimum edit distance, similarity keys, rules based on error-patterns, n-grams, probabilistic and soft computing techniques [Kukich, 1992]. The motivation behind the similarity key techniques and some of the rule-based techniques is to capture and correct phonetic errors. SOUNDEX [Odell and Russell, 1918] was the first similarity key technique, where the letters were mapped to 7 different phonetic code groups. *Phonetic code groups* are equivalence classes based on similar sounding or homophonous letters. Phonetex [Hodge and Austin, 2001], a sophisticated implementation of the same technique, have a larger number (14) of phonetic code groups and some simple phonetic transcoding rules. [Hodge and Austin, 2001] report that Phonix [Gadd, 1990], which uses language specific sophisticated phonology rules does not boost up the performance of the checker significantly. Moreover, language specific features render the system inflexible.

Van Berkel and DeSmedt [Van Berkel and DeSmedt, 1988] devised a spelling correction technique for Dutch surnames that first applies G2P conversion rules to the input word followed by a inverted-triphone dictionary lookup for retrieval of the correct spellings. The method was highly successful with 94% accuracy.

3.2 Spellcheckers for Partially Phonemic Languages

As stated before, most of the Indian languages are partially phonemic in nature and morphologically very productive. Therefore, we take a look at some of the works on Indian spellcheckers and the scope of improvement over correction schemes based on similarity key techniques. Out of the seven works presented on seven different Indian languages in a workshop on spellcheckers [TDIL, 2002], only one (“Detection and Correction of Phonetic Errors in Alphabetic Languages”, S. Bandopadhyay) explicitly deals with the issue of phonetic errors. The method is very similar to Phonetex and deals with only homophone substitution errors. Other works on Tamil, Telegu, Kannada and Bengali spellcheckers talk about the need of morphological analysis and generation for error detection and correction.

The main problems with the similarity key based methods are (1) they cannot handle context dependent variations²; and (2) the proper ranking of the suggestions are not possible. This can be illustrated by the Bengali vowel *a*, which can be mapped to /O/, /o/ or deleted depending on the context. A similarity key based technique might map this vowel to a phonetic code group θ everytime. Therefore, the Bengali word *tbaraNa*, pronounced as /tOron/, will be mapped to a code word³ $t\theta r\theta n\theta$. However, a word like *toraNa*, /toron/, will also get mapped to the same code. The nonword *taraNa*, whose graphemic edit distance from both *tbaraNa* and *toraNa* is 1, will be mapped to $t\theta r\theta n\theta$ and thus the spellchecker has no way to discriminate between the possible suggestions *tbaraNa* and *toraNa*. However, the fact that the nonword *taraNa* will be pronounced as /tOron/ in Bengali and not /toron/ provides enough reason to rank *tbaraNa* above *toraNa*.

A G2P that can distinguish between context dependent variations of *a* can be used to circumvent the aforementioned problem and properly rank the possible suggestions. It is this particular observation about partially phonemic languages that made us believe that the use of a G2P can certainly boost up the isolated-word error correction accuracy over the similarity key techniques.

4 The Spellchecker Model

The basic architecture of the Spellchecker proposed in this paper is shown in figure 1. The input word, which is in the graphemic form, is morphologically analyzed. If no valid morphological analysis is possible, the input is considered to be a nonword error and sent to the *isolated-word corrector* module. In the corrector module, first G2P is used to generate a *semi-phonetic* form of the input word, which is then searched for closest match against a structured semi-phonetic dictionary properly coupled with the corresponding graphemic word dictionary. The set of suggestions returned by the search algorithm are ranked by the ranking module to get the ordered list of suggestions.

4.1 Grapheme-to-phoneme Mapping

For phonemic and partially phonemic languages the set of graphemes Σ_G is identical to the alphabet set. The set of phonemes Σ_P is also language specific but can be subsumed by the universal set of

²Phonetex[Hodge and Austin, 2001] has overcome this problem to some extent by using simple context dependent letter-to-sound transcoding rules

³Assuming that the conjugate *tb* is properly mapped to /t/ by the phonetic transcoder.

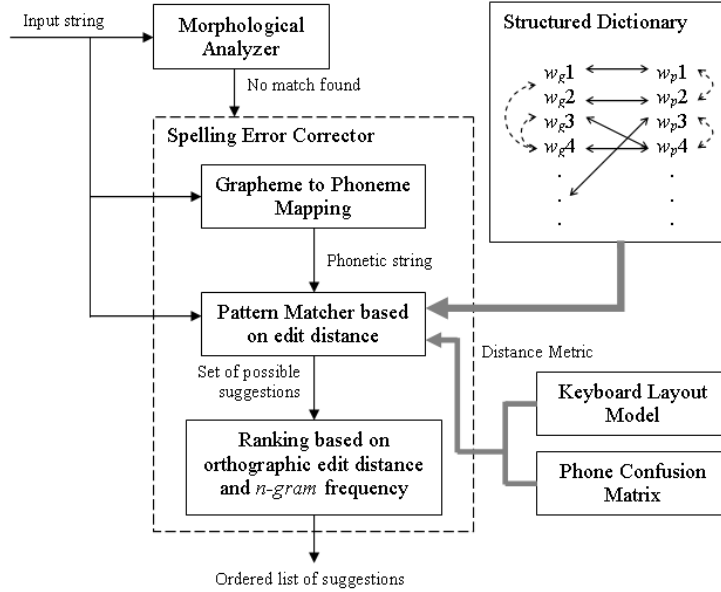


Figure 1: Architecture of the SpellChecker

phones. In its most natural interpretation, G2P for a language L can be defined as a map f_L such that

$$f_L : \Sigma_G^* \rightarrow \Sigma_P^*$$

Designing a complete G2P for languages which do not use strictly phonemic orthography is far from trivial and can be very resource intensive. However, a completely specified phonetic representation of a word is an overkill for a spellchecker. As discussed in section 3, most of the existing phonetic spellcheckers use either phonetic codes or very simple phonetic transcoding rules. For our purpose, we define a grapheme-to-phoneme mapping g_L for a language L as follows.

$$g_L : \Sigma_G^* \rightarrow (\Sigma_P \cup \Sigma_M)^*$$

Σ_M is a set of *metaphonemes* similar to phonetic codes, which usually correspond to a subset of Σ_P . Σ_M may also include a metaphoneme ϕ to represent the deletion of phonemes. It may not be possible to unambiguously determine the phonetic counterpart of a particular grapheme using simple rules, in which case the grapheme may be mapped to a minimal set of possible phonemes rather than a unique phoneme. Metaphonemes are nothing but a symbolic representation of such minimal sets. The phonetic code θ representing the set $\{/O/, /o/, \phi\}$, cited in section 3.3 is an example of a metaphoneme.

For strictly phonemic languages g_L can be very simply modelled using a simple function h_L such that $h_L : \Sigma_G \rightarrow \Sigma_P$. However, for partially phonemic languages, h_L is not sufficient for modelling g_L . More rules are required for specifying context sensitive features, which can be represented using *finite state transducers* [Kaplan and Kay, 1994]. Certain phenomena like schwa deletion in MIA cannot be

modelled using FSTs and might call for more sophisticated techniques [Choudhury and Basu, 2002]. In general g_L can be any computable function, which can take as input any arbitrary string over Σ_G . There is a tradeoff between the efficiency of g_L and the performance of the spellchecker.

4.2 Dictionary Structure

The dictionary can be considered as an undirected graph $\Delta(V, E)$. The set of nodes V is partitioned into two subsets V_G and V_P . Each node of V_G (V_P) represents a valid word in the graphemic(semi-phonetic) domain, i.e. a string over Σ_G ($\Sigma_P \cup \Sigma_M$). The set of edges E is partitioned into three subsets E_G , E_P and E_{G2P} , where

- $E_G \subseteq V_G \times V_G$ consists of numerically labelled edges $\langle v_{G1}, v_{G2} \rangle: i$ such that v_{G1} and v_{G2} are elements of V_G and the label i is a number equal to the graphemic edit distance between v_{G1} and v_{G2} .
- $E_P \subseteq V_P \times V_P$ consists of numerically labelled edges $\langle v_{P1}, v_{P2} \rangle: i$ such that v_{P1} and v_{P2} are elements of V_P and the label i is a number equal to the phonetic edit distance between v_{P1} and v_{P2} .
- $E_{G2P} \subseteq V_G \times V_P$ such that $\langle v_G, v_P \rangle \in E_{G2P}$ implies that a valid pronunciation of the word v_G is the semi-phonetic string represented by v_P .

The definition of E_G and E_P allows one to store the edit distances between any arbitrary pair of graphemic or phonetic words respectively, through properly labelled edges. However, for practical spellchecking applications one may need to search for words within a bounded edit distance and one need not store all the edges in E_G and E_P .

There are other standard dictionary compression techniques that can be applied on top of the aforementioned structure. Some of these will be discussed in the next section, but the spellchecking algorithm described here does not bank on those methods and the dictionary structure described above is both necessary and sufficient.

4.3 Pattern Matching using Edit Distance

The semi-phonetic representation of the word is searched for in the V_P using the standard edit distance based pattern matching algorithms (see [Navarro, 2001] for a detailed survey). For a spellchecker application, we need to appropriately modify the *penalty function*. The salient features of this modified penalty function are

- The penalty for substitution of a phoneme by a “similar sounding” phoneme like /r/ by /R/, or /s/ by /sh/ should be smaller than the penalty for substitution of say /r/ by /s/. The exact values for such phoneme substitutions can be arrived at by studying the spelling error patterns for a particular language. The substitution penalties between phonemes can be specified by a two dimensional matrix $DM_{phonetic}$, where the (i, j) th entry is the substitution penalty between phonemes i and j .

- The penalty for substitution by adjacent keys on the keyboard should be lower than substitution by spatially distant keys. Thus, the keyboard can be succinctly modelled using the penalty function, which can be specified as a two dimensional matrix $DM_{keyboard}$, where (l, m) th entry is the substitution penalty between graphemes l and m .
- The penalty for substitution of a metaphoneme by a phoneme that belongs to the subset it represents should be zero or in other words it is considered to be a perfect match rather than a substitution error.
- Penalties for insertion, deletion and swapping of characters can be again determined from studies on error patterns or assumed to be one as is generally done.

4.4 Generation of the Suggestion Set

The semi-phonetic representation of the input word can have an exact match with some node in V_P . In that case, the corresponding edges in E_{G2P} are traversed to find out the corresponding V_G nodes, which are included in the suggestion set. In other words, those words are retrieved which have the same pronunciation. The dictionary structure facilitates constant time retrieval of the homophonous words once the exact semi-phonetic word is found.

If no exact match is found, approximate matches within a fixed phonetic edit distance (say k) is looked for in V_P . Once such a node v_p is found, all the V_G nodes directly connected to that node through edges in E_{G2P} are included in the suggestion set. Next, for all the V_P nodes that are connected to v_p by edges in E_P labeled 1 to $2k$ are compared with the semi-phonetic input string and for every node searched, if the edit distance is within k , the corresponding V_G nodes for that V_P node are included in the suggestion set. The search is done up to edges labeled $2k$ to ensure that we do not miss any node in V_P whose edit distance from the input string is within k .

In order to capture typographic errors, the graphemic input string is also searched for approximate matches in V_G within a fixed orthographic edit distance (say k') in the same way. However, here we include any solution within the edit distance k' directly into the suggestion set.

4.5 Ranking the Suggestions

The suggestion set consists of elements which are within k phonetic edit distance and k' graphemic edit distance from the input. Initial ranking among the suggestions can be done by a weighted combination of phonetic and graphemic edit distance from the input word. Let

$$\lambda(w, w') = w_P \cdot d_P(w, w') + w_G \cdot d_G(w, w'), \quad w_P + w_G = 1$$

be the metric for ranking a suggestion w' , with respect to the input w and d_P (d_G) stands for phonetic (graphemic) edit distance. The smaller the value of $\lambda(w, w')$, the better is the suggestion. A higher value w_P penalizes a suggestion more for the non-phonetic errors, whereas a higher value of w_G penalizes a suggestion more for phonetic errors. Ideally, the values of w_P and w_G should be determined by experimental studies. Intuitively, for partially phonemic languages, w_P should be larger than w_G . For further discrimination among the suggestions one can use the n -gram statistics or user specific error modelling.

5 Implementation of a Bengali Spellchecker

Bengali is a partially phonemic language and displays a lot of interesting phonological, morphological and orthographic features. A Bengali spellchecker was implemented based on the approach described in this paper. The basic architecture of the spellchecker is very generic and language specific plugins and dictionaries can be used to develop spellcheckers for any other language.

5.1 Dictionary Precompilation

The graph like dictionary structure was precompiled from a Bengali wordlist by first applying G2P to obtain the corresponding semi-phonetic words. The semi-phonetic words are then partitioned into several files based on string length. The entries in each file are then alphabetically sorted in order to implement hashing and binary search techniques. A separate profile was automatically compiled for each of the semi-phonetic strings (nodes of V_P), which has pointers to the corresponding graphemic words (edges of E_{G2P}) and other semi-phonetic words up to a fixed edit distance $2k$ (edges of E_P). k was chosen as 1 for Bengali as the number of suggestions returned for $k > 1$ was very large and mostly inappropriate. Graphemic words were stored similarly and k' was also chosen as 1.

5.2 Some Issues Specific to Bengali

- Three metaphonemes were used for Bengali - θ_1 {/O/, /o/}, θ_2 {/o/, ϕ } and θ_3 {/O/, /o/, ϕ }. These were used whenever there was a confusion over vowel height assimilation or schwa deletion. The confusion between the phonemes /e/ and /E/ were kept unresolved and always mapped to /e/ (which in this case can be considered as the metaphoneme).
- Since no standard keyboard layout exist for Bengali, all the entries of the matrix $DM_{keyboard}$ were initialized to 1.
- All the entries of the matrix $DM_{phonetic}$ were initialized to 1 except for those corresponding to /r/-/R/, and a consonant and its aspirated form like /g/-/gh/ or /t/-/th/, which were initialized to 0.5. The choice of this value is arbitrary.
- Bengali has a productive agglutinative morphology. Echo words are often appended to the valid words (e.g. addition of nonsense word *Tala* after *jala* or *phala*). The spellchecker should not consider such words as spelling errors. Rules were plugged into the system to take care of such issues.

5.3 Performance

The spellchecker was tested on a set of manually generated nonwords. The error detection accuracy is 100%. However, it is difficult to grade the quality of suggested corrections. Qualitatively it can be said that for longer words with phonetic errors the suggestions were perfect, but for smaller words (length less than 6), the number of suggestions were large and a context dependent error correction technique seems to be absolutely necessary.

Currently, some of the inherent limitations of the system are: (1) errors at the word initial position other than phonetic ones are not handled; (2) the method is able to correct any number of phonetic errors in a word, but only a single typographical error other than the phonetic errors; (3) errors in inflected words are properly detected, but the system cannot suggest appropriate corrections unless the inflected word is also in the dictionary.

6 Conclusion

In this paper, an approach for developing spellcheckers for partially phonemic languages was introduced and a Bengali spellchecker based on this approach was described. The approach is generic and can be used for developing spellcheckers for other languages by incorporating some language specific plugins, like G2P rules and keyboard layout. Results from initial evaluation of the system are promising and a detailed quantitative evaluation is underway.

Some of the interesting issues that can be addressed in the future include analysis of spelling error patterns for partially phonemic languages, especially Indian languages; proper handling of agglutinative and fusional morphology while spelling error detection and correction; exploring and evaluating the different ranking techniques and context dependent word-error correction.

Acknowledgements

This work has been supported by Media Lab Asia research funding. We would also like to thank Prof Abhijit Das, Department of Computer Science and Engineering, IIT Kharagpur, for providing us with the Bengali dictionary that has been used for development of the Bengali morphological analyzer and the spell-checker.

References

- [Chatterji, 2002] S. K. Chatteji *The Origin and Development of the Bengali Language*. Rupa Co. New Delhi, INDIA, 2002.
- [Choudhury and Basu, 2002] M. Choudhury and A. Basu. A Rule-based Schwa Deletion algorithm for Hindi. *Proc. of the Int. Conf. on Knowledge-Based Computer Systems (KBCS)*, 343-353, Navi Mumbai, INDIA, 2002.
- [Gadd, 1990] T. Gadd. PHONIX: The Algorithm. *Program*, Vol. 24(4), 363-366, 1990.
- [Hodge and Austin, 2001] V. Hodge and J. Austin. An Evaluation of Phonetic Spell Checkers. *Technical Report YCS-2001-338(2001)*, Department of Computer Science, University of York, UK, 2001.
- [Kaplan and Kay, 1994] R. M. Kaplan and M. Kay. Regular Models of Phonological Rule Systems. *Computational Linguistics*, Vol. 20(3), 331-378, 1994.
- [Kukich, 1992] K. Kukich. Techniques in Automatically Correcting Words in Texts. *ACM Computing Surveys*, Vol. 24(4), 377-439, 1992.

- [Mitton, 1987] R. Mitton. Spelling Checkers, Spelling Correctors, and the Misspellings of Poor Spellers. *Information Processing and Management*, Vol. 23(5), 495-505, 1987.
- [Navarro, 2001] G. Navarro. A Guided Tour to Approximate String Matching. *ACM Computing Surveys*, Vol. 33(1), 31-88, 2001.
- [Odell and Russell, 1918] M. K. Odell and R. C. Russell. *U.S. Patent Numbers, 1,261,167 (1918) and 1,435,663 (1922)*. U.S Patent Office, Washington D.C., USA.
- [Pollock and Zamora, 1983] J. J. Pollock and A. Zamora. Collection and Characterization of Spelling Errors in Scientific and Scholarly Text. *Journal of American Society for Information Sciences*, Vol. 34(1), 51-58, 1983.
- [Pollock and Zamora, 1984] J. J. Pollock and A. Zamora. Automatic Spelling Correction in Scientific and Scholarly Text. *Communications of the ACM*, Vol. 27(4), 358-368, 1984.
- [TDIL, 2002] Indian Language Spell Checker Design Workshop. *viswabhaarat@tdil*. Technology Development for Indian Languages, 18-24, July 2002.
- [Van Berkel and DeSmedt, 1988] B. Van Berkel and K. DeSmedt. Triphone Ananalysis: A Combined Method for the Correction of Orthographical and Typographical Errors. *Proc. of the 2nd Applied Natural Language Processing Conference*, (Austin, Tex., Feb). Association for Computational Linguistics (ACL), 1988.
- [wikipedia, 2004] Anonymous http://en.wikipedia.org/wiki/Phonemic_orthography, 2004.
- [Yannakoudakis and Fawthrop, 1983] E. J. Yannakoudakis and D. Fawthrop. The Rules of Spelling Errors. *Information Processing and Management*, Vol. 19(2), 87-99, 1983.