Adaptive Language Independent Spell Checking using Intelligent Traverse on a Tree

Behrang QasemiZadeh
Iran University of Science and Technology
Tehran, Iran
QasemiZadeh@digitalclone.net

Ali Ilkhani
Digital Clone Corp.
Tehran, Iran
Ilkhani@digitalclone.net

Amir Ganjeii
Iran University of Science and Technology
Tehran, Iran
Amir@Ganjeii.com

***Abstract*—This paper introduces an adaptive, language independent, and 'built-in error pattern free' spell checker. Proposed system suggests proper form of misspelled words using nondeterministic traverse of 'Ternary Search Tree' data structure. In other words the problem of spell checking is addressed by traverse a tree with variable weighted edges. The proposed system uses interaction with user to learn error pattern of media. In this way, system improves its suggestions as time goes by.**

***Keywords*—Spell Checking, Ternary Search Tree, Learning and Adaptation , Error Pattern Modeling**

I. INTRODUCTION

The problem of detecting error in words and automatically correcting them is a great research challenge. Spell Checker systems have a vast application zone, e.g. internet search improvement [1] [2] [3], correction of errors caused by OCR[1] [4] [5], tools for text editors, Pre-processors for natural language processing, speech recognition, and etc [6].

The word-error can belong to one of the two distinct categories, namely, non-word error and real-word error [6]. Real-word error occurs when the usage of word, in relation with other words, sentence structure, or type of the text (Scientific, Sport, etc.) is not appropriate [6] [7]. Real-word error detection requires semantic analysis which is not the purpose of this paper. The focus, here, is mainly on the non-word error or misspelling. Misspells are detected when the specific word does not belong to the word domain of a language [6]. A spell checker system detects misspelled words, the position of errors, and it suggests the best similar word (s).

In order to detect the errors, it is necessary to model the related knowledge of words in a language for the system in either an explicit (Lexicon) [8] [9] [10] [11] [12] [13] [14] or an implicit (statistical models) way [5]. After error detection, it is necessary to specify possible types of errors and their correction methods for the system. This is generally accomplished by modeling the common error patterns.

Lexical Knowledge representation of a language is one of the significant issues in each system related to NLP [2]. Moreover, the lexical knowledge representation determines the general approach in system design and architecture. Lexicon's architecture can individually contain the implicit knowledge of the language. In general, computerizing the lexicon (dictionary) consists of parameters such as the size of dictionary [11], flexibility, the ability to generate all possible combinations [14], dictionary file structure, dictionary's segmentation, and techniques for words access [11].

Lexicons and their representations have been studied in details [11] [14]. Some researches have focused only on the lexicons containing roots of words. Morphological analysis has been utilized for the detection of the rest of words [4] [9] [10] [12]. In contrast, in some other researches all the words of language have been stored in the Lexicon and no lexical analysis is being utilized [8] [11] [14] [15]. The former approach is more complex, versus the latter, but it has a good measure of compression for knowledge representation.

Another important issue in designing the lexicon architecture is the search method of the words in lexicon. The most common method is using dictionaries with hash-tables structure [3] [13]. Its difficulties are proper definition of key for addressing, weak flexibility, and no compression of lexicon. N-gram is one of the frequently used methods for OCR. The most important issue of this method is the formation of a suitable graph of unprocessed text information [5]. The other common method for Lexicon representation is utilization of a tree based data structure [2].

Many researches have been done in order to model the error pattern and specifying its parameters. There are different categories for error patterns based on the source of errors. These categories are based on the structure of each language, pronunciation similarities [13], Typography (dictation

[1] Optical Character Recognition

[2] Natural Language Processing

similarity) [16], user's habits [12], and etc. Apart from the mentioned categories, the achieved pattern functions as a guide to detect error's place, hence fixing it. The main issue, here, is the dependency of error pattern to the language in which the system is running. Error pattern definition, regarding its dependence on language and media in which it use is time consuming, and is usually requires language's experts, even though, in most cases, these models are very accurate and efficient. Accuracy of the error pattern model has a straight effect on system's efficiency.

Dictation errors usually happen due to the following errors: (Figure 1) [2] [6]

- Substitution Error: Using a letter instead of the other.
- Deletion Error: Unintended elimination of one or more letters.
- Insertion Error: Unintended insertion of a letter in a word.
- Transposition Error: Transposition of two adjacent letters.
- Split Word Error: Attaching two correct words.

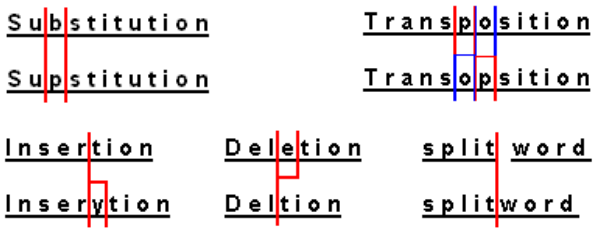

Figure 1. Usual dictation errors.

Thus, each suggestion of system for a misspelled word is derived from applying one or more of the changes mentioned above on the input string. According to what was stated, in order to propose the proper suggestions, the spell checker faces a vast search space where only one word, among the suggested ones, should be selected as the correct one. In spell checker system, one of the important goals is to limit the search space, with the help of error pattern models, in order to suggest the best similar words with the optimal search and the least computational cost [16].

In order to achieve the main goal of spell checker, which is error detection and correction, it is needed to store a proper integration between Lexicon and the structure of error pattern models.

Another important issue in spell checker design is whether a spell checker system has an interaction with a user or not. In the latest systems, it is assumed that the spell checker is used in a user interactive environment [13] [15] [16]. The system prepares a list of suggested words from where the user can make the final choice. In some others, according to their application, for example as a post-processor for an OCR system or a Speech to Text system, the spell checker proposes only one suggestion without any interaction with a user [4] [5] [8].

The remaining parts of this paper include the followings: Section 2 reviews some related works. The proposed method is introduced in Section 3. Experimental results are discussed in section 4. Finally, section 5 concludes the work.

II. Related Works

Spell checking has a long history in Computer science [17]. The proposed methods consist of Edition Distance (ED) [2] [13] [16], rule-based techniques, probability techniques [7] [15], n-grams models [4] [5], expert systems [14], similarity key methods [13], and hybrid methods [8] [6] [10]. In most of these methods, the first step is to prepare language-related lexicons and extracting the error patterns. In the next step, the error patterns will be modeled in order to detect the position of the errors and propose the suitable error-removal solution. The outputs of such systems are usually a list of the most similar proper words based on error models [6] [10] [15] [16].

The algorithms, based on the least ED, normally define the ED with a determined function. The System suggests words with the least ED for the given misspelled word as its suggestion [6] [16]. In this method, the error pattern is modeled by parameters of a distance function. The accuracy and speed of the algorithm depends on the definition of the ED and can be flexible depending on its definition.

Similarity key methods, such as SoundEX systems and Metaphone algorithms [13] try to propose a map between similarity keys and specification of words [10]. In this method, 'Hashing' structure is often used to represent Lexical knowledge. Map function has also the role of addressing [13]. This method uses the parameters of map function to model error patterns. Because of the hash table structure, the accuracy and speed of similarity key methods depend on key's definition (error pattern). This method has suitable accuracy when the error pattern model is properly defined.

In n-gram based methods, the occurrence probability of characters stream of a word is calculated. Lexical knowledge in n-gram methods is represented in an implicit way. It tries to model the words of language statistically. This method is usually used as a post-processor to achieve far better results with OCR applications [4] [5].

In [12], an adaptive architecture is described for a spell checker. The system adapts itself with user, using different order of words in the list of suggested words. The error patterns of language are predefined in different knowledge bases. In fact, the error patterns model of the system is fixed and it can not be changed.

III. The Proposed Method

Our method suggests an adaptive, language independent spell checking tool. It is based on 'Ternary Search Tree' (TST) data structure. The proposed method learns media error pattern and improves its suggestions as time goes by. Instead of using expert knowledge for error pattern modeling, this method learns error pattern by interaction with user.

Figure 2 shows the general scheme of the proposed system. Spell Checker consists of five parts: Spell Checking Module

(SCM), Lexicon, Cost of Transition (COT), Learning and Adaptation Module (LAM), and more importantly a user. The role of SCM is detection of errors and proposing proper suggestions. The role of LAM is to learn media error pattern by interaction with user. Error pattern has been implicitly modeled in COT. Lexicon contains the words of language in a TST data structure. Furthermore, two threshold limits are used in order to control and restrict the search space size when suggesting proper word. 'Global Threshold' limits the number of suggested words in a single suggestion entry, while 'Local Threshold' makes a limitation for each alteration in suggested word(s) components of a single suggested entry.

SCM takes an input stream. If the word does not exist in the Lexicon, words with minimum path cost in TST based on COT would be suggested as a proper form of a misspelled word. List of extracted words with traversed paths in TST will be sorted by total path cost. It is then stored in the suggestion list and is sent to the user. Due to user selection, input stream would be added as a new word to Lexicon or COT would be updated. Figure 2 explains data flow diagram between these modules.

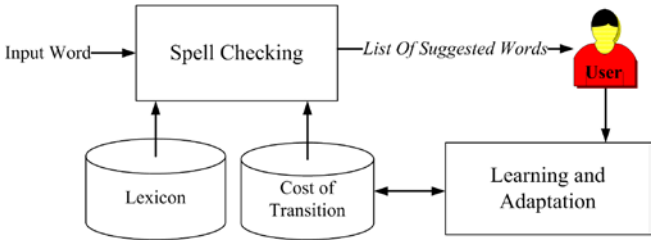

Figure 2. Modules and relations between them.

The system components are explained below in details.

A. *Lexicon*

Lexicon represents the lexical knowledge of language. TST data structure with weighted edges is used to represent lexicon. TST data structure was introduced in [19]. Here, each node of TST stores a single character. Each node points to three other nodes, one to the left, one to the right, and one in the middle of them. The left pointer points to a letter with a smaller character code, while the right pointer points to a letter with a greater character code. The middle pointer points to the next character in the input stream. The tree traverse in the left or right nodes does not cause the traverse in the input stream. This data structure compresses the data with the same prefix. Frequent prefixes are saved only once.

Modified TST structure in the proposed method includes the assignment of costs to edges of tree and a flag which displays the end of word. The costs related to each edge are categorized and have been saved in an individual data structure named 'Cost Of Transition', in order to decrease the volume of data structure and the facility in adaptation process. Addition of cost to edges of the tree causes the change in traverse algorithm of the tree. It will change the traverse from a classic procedure to a non deterministic one.

B. *Cost of Transition*

As mentioned above, all similar transitions have similar costs. They are saved in a matrix based data structure. The edges of tree are categorized due to their starting and ending characters. Each row shows the starting character and each column shows the ending character. Values of matrix cells show the weight of the specified edge, in the other words, cost of transition between two characters.

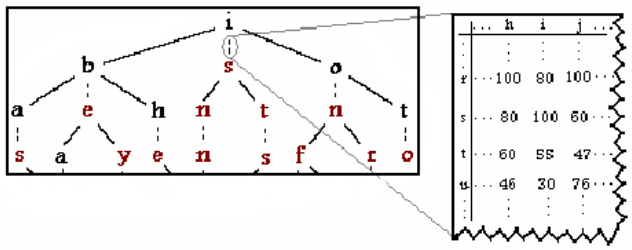

Figure 3. Lexicon and representation of weighted edge in COT matrix.

In order to add the deletion and insertion operating ability and adding learning ability for them, a 'Null' column and row is added to the COT matrix to save the cost of deletion and Insertion. As an example, for Persian, COT matrix could be a 40*40. 28 of rows and columns are labeled with English letters and the rest are labeled with other common symbols that are used in writings. Figure 3 reveals a part of this matrix. At the beginning, the entire matrix cells have the value of 100.

C. *Spell Checking*

As previously mentioned, according to the specified architecture, the spell checking process has been transformed into non-deterministic traverse of a tree with weighted edges. In other words, the problem of spell checking is transformed to the problem of traversing a weighted tree with minimum total weight. With the help of tree traverse, Spell Checking module provides user a list of suggested correct words. The search process for the tree representation of lexicon, in order to present the suggestions list, is as following:

- If the character in the current node of the tree is the same as the character in the input stream, it will be traversed on the stream and tree. Otherwise, current character in input stream will be transformed into the saved character in tree's current nodes, according to the specified cost in COT. This transformation and its related cost will be saved. In the case that the current node of the tree represent the end of a word, the suggested word will be saved, and by looking up the tree root, it will be tried to traverse the rest of the stream, with the condition that the sum of implemented costs does not cross the 'Local Threshold' and 'Global Threshold'. If the current node is not at the end of the word, the traversing of remaining stream will be processed from the middle pointer.
- If the character in the current node of the tree is different from the current character in input stream, and no relocation has taken place, and the relocation of the input stream current character with the next character is possible, then this relocation will take place and the remaining of the tree will be traversed.
- If the character in the current node of the tree is different from the current character of the input stream,

and it is possible to transform current character of the input stream into a null character according to the exploited cost from COT, 'Local Threshold', and 'Global Threshold', then the current character will be deleted and the tree traverse from the current node will be continued, according to the input stream's next character.

- If the character in the current node of the tree differs from the input stream's current character, and it is possible to transform the invalid character in the input stream into character of current node according to the exploited cost from COT matrix, 'Local Threshold', and 'Global threshold', then the character of current node will be inserted and the tree traverse will be continued according to the current character of the input stream and from the middle node.

The mentioned traverse will be also examined for the left and right nodes. This operation will be continued until reaching the end of the input stream or an invalid node.

D. Learning and Adaptation

The role of LAM is to modify the cost of transition of tree. In other words, the role of LAM is to learn media error pattern (such as user's habit for dynamic media or OCR problems for static media and so on) and to add new words to Lexicon. If input string does not exist in Lexicon, it would be detected as a misspelled word and could be added to Lexicon by standard insert function of TST data structure. If user selects one of suggested words of system, this choice causes the change in cost values and weights of tree edges in order to decrease the cost of selected suggestion and increase the cost of other suggestions for same misspell word. Cost values in COT will be calculated by the (1), if the user's selection is not the first suggestion in the list.

$$COT(C_i,C_j)=\begin{cases} COT(C_i,C_j)_{old} * (1-\alpha * index) & index = selected \\ COT(C_i,C_j)_{old} * \left(1+\dfrac{\alpha}{index}\right) & index \neq selected \end{cases} \quad (1)$$

Where α is the learning rate, and index is the number of suggestions in suggested list.

E. Thresholds

To limit the search space when traversing the tree, two thresholds namely Local and Global Thresholds are used. Local Threshold limits search depth and Global Threshold prevent from generating unsuitable consequence of words.

Local threshold definition is based on the length of each suggested word. Global threshold definition is based on the length of input string and the number of words in each suggestion entry. If the length of suggested words is shown by Ls, the length of input string is shown by Li, and Ni shows the number of words in a suggestion entry then a simple formula for Local and Global thresholds can be defined as (2), and (3).

$$LocalThershold(L_s)=\lambda * \left(\begin{cases} 2 & \ln(L_s)<2 \\ \ln(L_s) & otherwise \end{cases} \right) * \exp^{-(N_s-1)} \quad (2)$$

$$GlobalThershold(L_s)=Minimum(LocalThershold(L_i),\gamma * Min_s) \quad (3)$$

Where λ is the maximum allowed dissimilarity rate to control the depth of search and γ is a limitation rate for the number of suggested word.

IV. Experimental Result

The proposed system has been tested for two languages, Persian and English. In Persian, since there was no data-set to test the system, we gathered a list of misspelled words. The list contains 5595 misspelled words and their correct forms. In English, we have used common misspelled words which have been used to test other spell checker systems before[3]. English, data-set contains 547 words. For both English and Persian, test-bench has been randomly divided into two parts: one for training and the other for system testing.

For Persian, 3827 words were spotted as training set and 1768 were selected for testing. In this case lexicon contained 40000 Persian words. As English, 360 words were spotted as training set and 187 were selected for testing. In this case lexicon contained 25000 English words

The precision of system was calculated based on (4). The presented formula for precision is different from its classic form due to the importance of the index of words in the list of suggestions.

$$\Pr ecision = \sum_{i}^{N} \frac{Index_i}{10 * N}$$
$$Index_i = \begin{cases} 11 - SuggestNo_{w_i} & 1 \le SuggestNo_{w_i} \le 10 \\ 0 & Otherwise \end{cases} \quad (4)$$

Hear 'N' stands for the total number of words in test/train set, 'w_i' indicates the i^{th} word in the suggestion list and '$SuggestNo_{wi}$' is the selected word's index in the suggestion list for the word number 'i' in the test/train set.

For both of these languages, at first, COT matrix cells are initiated by 100. The learning rate, α, is set to 0.01. Also λ, Maximum allowed dissimilarity rate, and γ, limitation rate for the number of suggested word, are set to 35 and 1.50.

For each data-set, three figures represent the results. Figure 4 and 7 show how many words are suggested correctly as the first word in the suggestion list. Figure 5 and 8 show the precision of system in each iteration based on (4). The number of misspelled words which have no suggestion is shown in figure 6 and 9. They can be interpreted as a measure of system coverage.

[3] http://www.aspell.net/test/

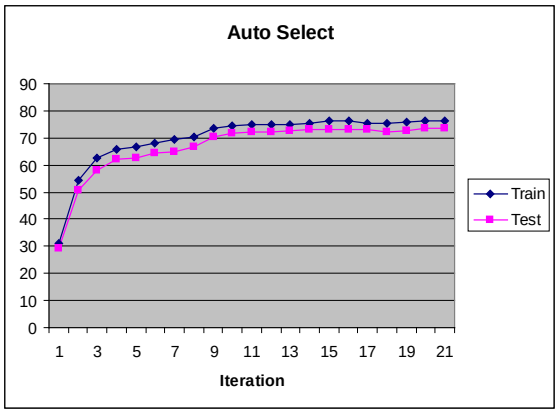

Figure 4. Auto-Select Precision (Percentage/Iteration) for Persian data-set.

As figure 4 and 5 show, since the sources of errors are various, the change in learning level is not noticeable after certain learning iterations. In other words, even though learning of error pattern, itself, results in emergence of other errors in the whole data-set, it can cause improvement in some other errors e.g. precision. Similar problem happens for English data-set as shown in figure 7 and 8.

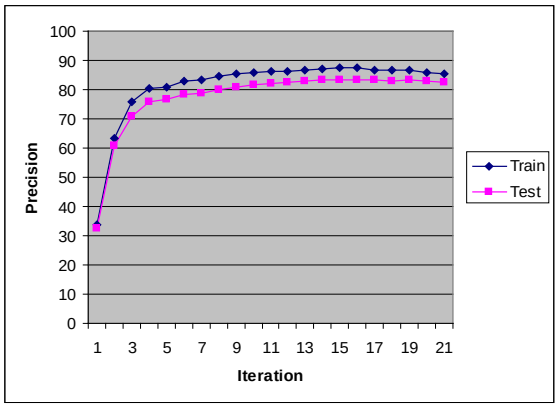

Figure 5. System Precision (Percentage/Iteration) for Persian data-set.

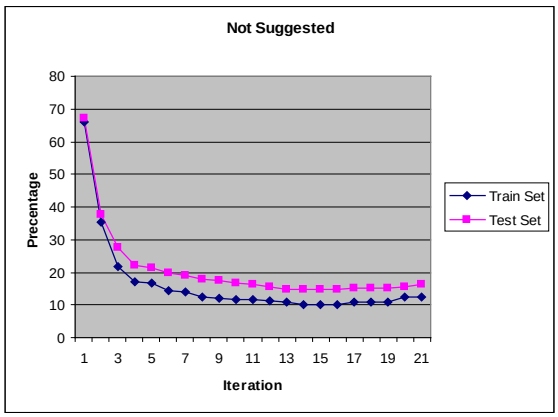

Figure 6. No Suggestion (Percentage/Iteration) for Persian data-set.

Figure 6 and 9 show the number of words which system failed to generate any suggestion for them. As previously mentioned, after certain iterations, 'error pattern learning' for some of words results in system's inability to suggest correct form of some other words which system was previously able to provide a suggestion list for them.

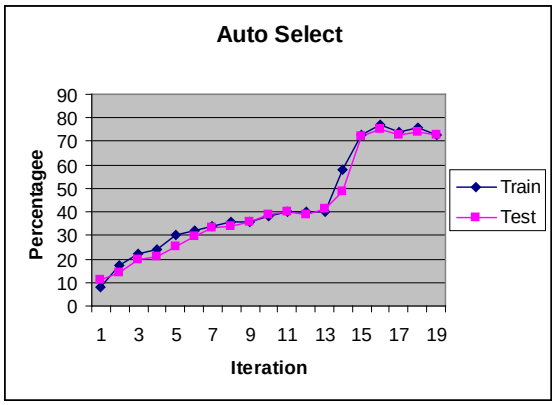

Figure 7. Auto-Select Precision (Percentage/Iteration) for English data-set.

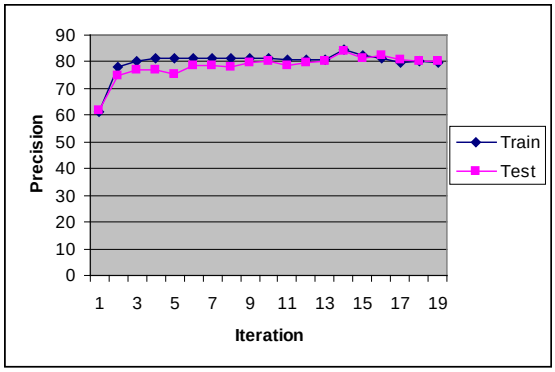

Figure 8. System Precision (Percentage/Iteration) for English data-set.

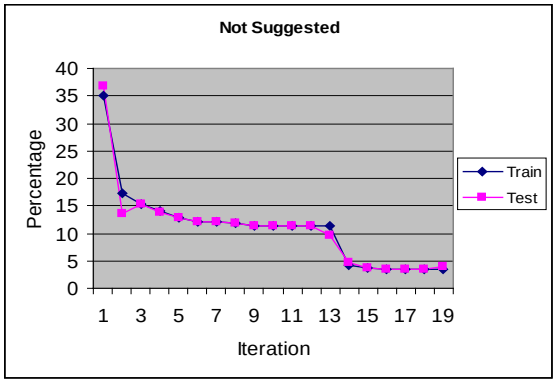

Figure 9. No Suggestion (Percentage/Iteration) for English data-set.

V. CONCLUSION

This paper introduced a novel 'language independent' spell checker system which can learn the media error pattern with some sample from a language or a media in which the system is used. The proposed method can adapt itself by interactions with user or outer media. it improves its suggestion list as time

progresses. In other words, the system can learn the error pattern of user and media.

One of the most important issues in designing a spell checker system is the popper definition of media error pattern. It is due to the fact that, it has a great influence on the results of spell checker system. Generally, error patterns are fixed and language experts define these patterns. This resulted in a language dependent system which can not be used for other languages. Such a system could not be used to detect misspelled words in other media except for the media that has a previous defined error pattern. The error pattern learning ability of the proposed method can overcome this shortcoming.

In the proposed system, adding a new language is equal to adding a lexicon and a COT matrix; therefore, the localization of the proposed system is easy. The system will extract the common error patterns without the help of experts; In other words it learns new media/language error pattern by some samples. Briefly, the introduced approach has more flexibility, more data compression rate, and more accuracy in comparison with other proposed methods. However our method is sensitive to media, but it is shown that it has acceptable results for auto-selection problems. For future work, we have decided to use language models to improve the accuracy of system.

REFERENCES

[1] H. Dalianis, 'Evaluating a spelling support in a search engine', In Proceedings of NLDB-2002, the 7th International Workshop on the Applications of Natural Language to Information Systems, June, 2002.

[2] Bruno Martins, Mário J. Silva, 'Spelling correction for search engine queries', EsTAL, pp. 372-383, 2004.

[3] Mansour Sarr, 'Improving precision and recall using a spellchecker in a search engine', Master's Thesis in Computer Science, Stockholm University, 2004.

[4] Sait Ulaş Korkmaz, G. Kırçiçeği Y. Akıncı, Volkan Atalay, 'A Character recognizer for turkish language', Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR), IEEE, 2003.

[5] Li Zhuang, TaBao, Xiaoyan Zhu, Chunheng Wang, Satoshi Naoi, 'A Chinese OCR spelling check approach based on statistical language models', International Conference on Systems, Man and Cybernetics, Hague, Netherlands, pp. 4727-4732, IEEE, Oct. 2004.

[6] Bidyut Baran Chaudhuri, 'Reversed word dictionary and phonetically similar word grouping based spell-checker to bangla text', Indian statistical Institute, Kolkata, India, 2000.

[7] Filip Ginter, Jorma Boberg, Jouni J¨arvinen, Tapio Salakoski, 'New techniques for disambiguation in natural language and their application to biological text', Journal of Machine Learning Research 5, pp. 605-621, 2004.

[8] Dustin Boswell, 'Language models for spelling correction', CSE 256, Spring 2004.

[9] Johan Carlberger Rickard, Domeij Viggo Kann Ola Knutsson, 'A Swedish grammar checker', Association for Computational Linguistics, 2000.

[10] T Dhanabalan, Ranjani Parthasarathi and T.V. Geetha, 'Tamil spell checker', Sixth Tamil Internet 2003 Conference, Chennai, Tamilnadu, India, August 22-24 2003.

[11] Boubaker Meddeb Hamrouni, 'Logic compression of dictionaries for multilingual spelling checkers', Proceedings of the 15th conference on Computational linguistics, Kyoto, Japan, August 05-09 1994.

[12] Menno van Zaanen, Gerhard van Huyssteen, 'Improving a spelling checker for Afrikaans', Language and Computers, Publisher Rodopi, ISSN 0921-5034, vol. 47, no. 1, pp. 143-156, August 2003.

[13] Arif Billah Al-Mahmud Abdullah, Ashfaq Rahman, , 'A Generic spell checker engine for south asian languages', accepted for publication and presentation at the IASTED 2003 refereed conference on 'Software Engineering and Applications' ~SEA 2003, Marina del Rey, CA, USA'. Paper No 397-045, November 3-5 2003.

[14] Sandor Dembitz, Petar Knezevic, Mladen Sokele, 'Developing a spell checker as an expert system', Journal of Computing and Information Technology - CIT 11, pp. 285–291, 2004.

[15] Deepak Seth, Mieczyslaw M. Kokar, 'SSCS: a smart spell checker system implementation using adaptive software architecture', IWSAS, pp. 187-197, 2001.

[16] Per-Ola Kristensson, Shumin Zhai, 'Relaxing stylus typing precision by geometric pattern matching', ACM Conference on Intelligent User Interfaces (Proc. IUI 2005), ACM Press, pp. 151-158, 2005.

[17] K. Kukich, 'Techniques for automatically correcting words in text', ACM Computing Surveys, 24(4):377.440, 1992.

[18] Gerasimos Potamianos, Frederick Jelinek, 'A Study of n-gram and decision tree letter language modeling methods', Speech Communication 24, pp. 171-192, 1998.

[19] Jon L. Bentley and Robert Sedgewick, 'Fast algorithms for sorting and searching strings', Proceedings of the eighth Annual ACM–SIAM Symposium on Discrete Algorithms, pp. 360–369, January 1997.

[20] Loghman Barari and Behrang Qasemizadeh, CloniZER Spell Checker: Adaptive language independent spell checker, ICGST, 2005, Cairo.