www.icgst.com

CloniZER Spell Checker
Adaptive, Language Independent Spell Checker

Loghman Barari[1] Behrang QasemiZadeh[1]
Digital Clone, Speech and Language Department,
6th floor, No 880, College cross, Enqelab Ave, Tehran, Iran
[Barari, QasemiZadeh]@DigitalClone.net,
http://www.digitalclone.net

Abstract
CloniZER spell checker is an adaptive, language independent and 'built-in error pattern free' spell checker tool which is based on 'Ternary Search Tree' data structure. It suggests the proper form of the misspelled words using nondeterministic traverse. In other words the problem of spell checking is addressed by traverse a tree with variable weighted edges. The proposed method learns media error pattern and improves its suggestions as time goes by. Instead of using expert knowledge for error pattern modelling, the proposed algorithm learns error pattern by interaction with user.

Keywords: *Spell Checking, Error Pattern, Learning and Adaptation, Lexicon, Ternary Search Tree (TST).*

1. Introduction

Nowadays, having general access to Internet as a universal phenomenon, electronic texts development and using the text based query interfaces have made the existence of assisting tools for text manipulation inevitable. For example, about 10 to 12 percent of the queries entered in search engines, have dictation errors [1]. Spell Checker have a vast application zone, such as internet search improvement [2][3], correction of errors caused by OCR [2] [4][5], tools for text editors, Pre-processors for natural language processing, speech recognition, and the pen-based computer interfaces.

There are two possibilities for errors: namely non-word error and real-word error [6]. Real-word error occur while the usage of word, in relation with the previous or next word, or, sentence structure and type of the text (Scientific, reportorial, etc) is not appropriate [6][7]. Real-word error detection is in need of high level semantic information, type of word ambiguity detection, and its multiple applications, which is not the purpose of this document. The focus, here, is mainly on the non-word error or misspelling. Misspells are detected when the specific word does not belong to the language words domain [6]. The duty of spell checker is to detect the position of errors and suggests the best similar word (s).

In order to detect the errors, it is necessary to model the related knowledge of language words for the system, in either an explicit (Lexicon) [8] [9] [10] [11] [12] [13] [14] or implicit way (statistical models) [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 common errors patterns modeling.

Knowledge representation of words of a language is one of the significant issues in each system related to NLP [3]. Moreover, knowledge representation of words determines the general approaches in system design and architecture. Lexicon's architecture can individually contain the implicit knowledge of the language. In general, computerizing the dictionary consists of parameters such as the size of dictionary [11], flexibility, the ability of generating all possible combinations [14], dictionary file structure, dictionary's segmentation, and word access' techniques. [11]

In some researches, Lexicons and their representations have been studied in details [11] [14]. Some researches have focused only on the Lexicons containing stems of words, and inflection rules and morphology have been utilized for the detection of the rest of words [4][9][10][12], versus others, where whole words of language have been presented in the Lexicon and no lexical analysis is being utilized [8][11][14][15]. The former approach is more complex, comparing to the latter one but it has a good measure of compression for knowledge representation. Another important issue in designing the lexicon is the search method and access to the words. The most common method is using dictionaries with hash-tables structure [3] [13], which its difficulties can be named as proper definition of key for addressing, weak flexibility, and no compression of lexicon. N-gram is one of the frequently used methods for OCR which the most important issue of this method is

[1] Digital Clone Corp. & Iran University of Science and Technology
[2] Optical Character Recognition
[3] Natural Language Processing

the formation 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] and Typography (dictation similarity) [16] user's habits [12] and etc. Apart from the mentioned categories, the achieved patterns can function as a guide to detect error's place and fix it. The main issue, here, is the dependency of error pattern to the language in which the system is running. Error pattern detection, regarding its dependence on language and media in which it use, is though, time consuming, and is usually in need of language's experts, although, in most cases, these models are very accurate and efficient. Accuracy of the achieved error model has a straight effect on system's efficiency.

The cause of usual dictation errors can be categorized as follow: (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 separate words.

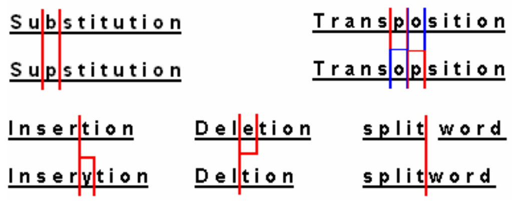

Figure 1. Usual Dictation Errors

Thus, each system's suggestion for a wrong word is derived from applying one or more of these changes mentioned above on the input string, although in the current spell checker systems, not all the above errors are covered. According to what was previously stated, in order to propose the proper suggestion, the spell checker faces a vast search space, where only one word, among the suggested ones, should be selected as the proper word. 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 word, with the optimal search and least computational cost [16].

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

The remainder of the paper is organized as follow: Section 2 reviews some related works, in Section 3, proposed method has been introduced, the. Experimental results are shown in section 4. Finally Section 5 points our conclusion.

2. Related Works

Spell checking has a long history in Computer science [17], and nowadays spell checking system is as an essential part for almost all application software [14]. The proposed methods in these systems consist of Edition Distance (ED) [2][13][16], rule-based techniques, probability techniques [6] [15] and n-grams models [4][5], expert systems [14], similarity key methods [13], and hybrid methods [6][8] [10]. In most of these methods, the first step is to prepare a language-related lexicon and error pattern extracting. In next step, error patterns will be modeled, in order to detect the errors position and proposing the suitable error-removal solution. The output of such systems is usually a list of 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 word with the least ED with the given word will be picked up as a winner and system will suggest the winner word [6] [16]. For example, a very simple ED function can be defined by appropriating weight to each change that must be applied, in order to alter the word to its correct mode, such as characters Insertion or deletion and simple mathematic operations [6][13]. In this method, ED definition, such as the number of operations and their weight will have a direct effect on the algorithm accuracy. In other words, the error pattern is modeled by the distance measure function parameters. In these methods, the accuracy and acceleration of the algorithms depends on the definition of the ED, and can be flexible, depending on its definition.

Similarity key methods try to propose a map between word and key, according to its features and heavily depended on error pattern model [10] such as SoundEX systems and Metaphone algorithms [13]. In this method, 'Hashing' structure is often used to propose Lexicon knowledge, and map function has also the role of addressing [13]. In other words, the parameters of map function are used for error pattern modeling. Because of hash table structure, similarity key methods accuracy and speed depends on key's definition. This method has suitable and high 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. If "s" is a character sequence, and h_i is the whole history information before i^{th} word "w_i", then the probability of "s" will be calculated by [18]:

$$P(s) = P(w_1 w_2 \ldots w_t) = \prod_{i=1}^{t} P(w_i | h_i) \quad (1)$$

Now, if this history is limited to n-1 characters, (1) will be changed to (2):

$$P(s) = \prod_{i=1}^{t} P(w_i | h_i) = \prod_{i=1}^{t} P(w_i | w_{i-n+1} w_{i-n+2} \ldots w_{i-1}) \quad (2)$$

Lexicon knowledge in n-gram methods is implicit and tries to model the words of language statistically. This method is usually used as a post-processor in OCR [4] [5].

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

3. CloniZER Spell Checker

CloniZER spell checker is an adaptive, language independent and 'built-in error pattern free' spell checking tool, which is based on 'Ternary Search Tree' (TST) data structure. it suggests the proper form of the misspelled words using variable cost of traverse. The proposed method learns media error pattern and improves its results as time goes by. Instead of using expert knowledge for error pattern modeling, this method learns error pattern by interaction with user.

Figure 2 is the general scheme of the proposed system. Proposed system consists of five parts: Spell Checking Module (SCM), Lexicon, Cost Of Transition (COT), Learning and Adaptation module (LAM) and finally a user. SCM's role is to detect the errors and propose 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 words of language in TST data structure. Furthermore, two threshold limits are used in order to control and restrict the search space size in proper word suggestion. “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 or words components of a single suggestion entry.

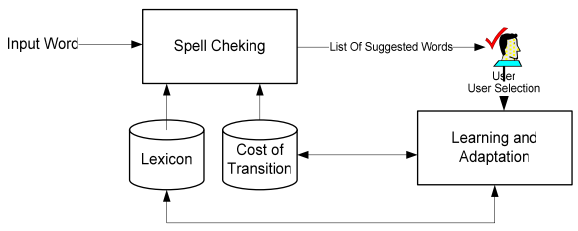

Figure 2. CloniZER Spell Checker Modules

The input stream is received by Spell Checker. If the word does not exist in Lexicon, words with least path cost in TST, based on COT, would be suggested. List of extracted words with traverse path in TST will be sorted by total path cost and hold in suggested list and sent to user. Due to user selection, input stream would be added as a new word to Lexicon or COT would be updated. Figure 2 explained data flow diagram between these modules.

Each system’s component will be explained in the following in details.

3.1 Lexicon

Lexicon represents the lexicon knowledge of language in TST data structure with weighted edges. The TST has been first introduced in [19]. This data structure compress data with same prefix and frequent prefix will be saved only once. In TST data structure, in each node of tree, a character will be saved. Each node points to three other nodes in left, middle and right. The left pointer, points to the letter with smaller code, while the right pointer points to a letter with bigger code. The middle pointer points to the next character in input stream. (Figure 3) The tree traverse in left or right nodes will not because the traverse in input stream.

Modified TST structure in CloniZER Spell Checker contains the addition of weight to edge of tree and a flag which displays the end of word. The weight related to each edge are categorized and has 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. Additions of cost to edge of the tree cause the change in traverse tree algorithm, which has changed the traverse from a classic procedure to a non deterministic one.

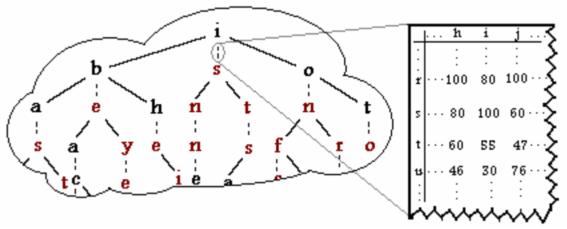

Figure 3. Lexicon and COT

3.2 Cost of Transition

As mentioned above, all similar transitions have similar costs, which save 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, value of matrix cells hold weight of the specified edge, in the other words cost of transition between two characters.

In order to add the deletion and inserting operation 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. For example, for Persian, COT matrix is a 40*40 square one which 32 of rows and columns are labeled with Persian letters and the rest are labeled with other common symbols that are used in Persian writings. Figure 3 reveals a part of this matrix. At the beginning, the entire matrix cells have the value of 100.

3.3 Spell Checking

As already 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, in order to present the suggestions list, is as following:

A. Transposition Error:
   a. If the stored character in current node is the same as the character in the input string, it will be move pointer on input string and traversed on tree. Otherwise, current character in input stream will be transformed into the stored character in tree's current nodes, according to the specified cost in COT, and this transformation and its related cost will be held.
   b. In case the current nod is at the end of the word, the suggested word will be held, and by returning to the root of tree, it will try to traverse the rest of the string, with this condition that the sum of implemented costs does not cross the "Local Threshold" and "Global Threshold".
   c. If the current node is not at the end of the word, the traversal of remainder of the stream will be processed from the middle pointer.

B. Substitution Error:
   a. If the stored character in the current node is different from the current character in input string,
      i. If the relocation of the input stream current character with the next character is possible and no relocation has taken place, this relocation will take place, and the remainder of the tree would be traversed.

C. Insertion Error:
   a. If the possibility of the transformation of input stream's current character into a null character, according to the exploited cost from COT and its comparing with "Local Threshold” and "Global Threshold" exists, the current character will be deleted and the tree traverse from the current node would be continued, according to the input stream's next character.

D. Deletion Error:
   a. If the possibility of the transformation of null character, into current node's character, according to the exploited cost from COT matrix and its comparing with "Local Threshold" and "Global threshold" exists, the current node's character will be inserted and the tree traverse will be continued according to input stream's current character and from the middle node.

E. Cover other possibilities:
   a. The mentioned traverse will be examined for the left and right nodes. This operation will be continued until reaching the end of the input stream or an invalid node.

3.4 Learning and Adaptation

The role of LAM is to modify the cost of transition and the weight of edges of tree. On the 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 following formula if the user's selection is not the first suggestion in the list:

$$COT\,(C_i, C_j) = \begin{cases} COT(C_i, C_j)_{old} * (1 - \alpha * index) & index = selected \quad (3.1) \\ COT(C_i, C_j)_{old} * \left(1 + \dfrac{\alpha}{index}\right) & index \neq selected \quad (3.2) \end{cases} \quad (3)$$

Where α is the learning rate (0.1>α >0), and index is the number of suggestion in suggested list. One suggestion for α is (4):

$$\alpha = 0.1 * \frac{1}{L_{SL}} \quad (4)$$

Where L_{SL} is the length of final suggestion list.

As mentioned before, according to (3.1) if user selected word is not in top of list, all cost for traversing tree to reach this suggestion string would be reduced according to its position in the suggestion list; in other words when index of selected words tend to bottom of list its related cost would strongly be reduced, so that learn the error pattern be faster. In other hand, for miss-suggestion string, their cost would be increased according to their index in list. Cost of miss-suggestion in top of list strongly increased in comparing to miss-suggestion in bottom of list.

For practical implementation, it’s better to bound cost of transition value; here it is limited to (5~100).

3.5 Thresholds

Another important issue in CloniZER spell checker is Local and Global Thresholds which bounded TST traverse search space for extracting similar words. 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 $L_{suggest}$ and the length of input string is shown by L_{input} and N_{words} is the number of words in suggestion entry then a simple formula for Local and Global thresholds can be defined as follow: (5), (6)

$$GlobalTher\ shold\left(L_{suggest}\right)= \\ Minimum\left(LocalThers\ hold\left(L_{input}\right), \gamma * Min_{suggests}\right) \quad (5)$$

$$LocalThers\ hold\left(L_{suggest}\right)= \\ \lambda * \left(\begin{cases} 2 & \ln\left(L_{suggest}\right) < 2 \\ \ln\left(L_{suggest}\right) & otherwise \end{cases}\right) * \exp^{-\left(N_{words}-1\right)} \quad (6)$$

Where λ is Maximum allowed dissimilarity rate for bounding the search space and γ is suggest word count limitation rate (γ>1).

When λ tends to be the maximum cost, local threshold, search space, process cost and the number of suggestions will be grater, vice versa. According to our test, simple assumption for λ could be 30% of maximum cost.

γ is bounded the search space in other way. In other words, γ is acceptation rate and it rejects suggestions which cost of new suggestions is too much as best later found suggestion. According to our test, the value more than 2 for γ has no effect on global threshold.

4. Experimental Result

Unfortunately there is no standard database for Persian spell checking evaluation. In order to test the system, a data set has been prepared from two different sources: (Table 1)

1. The incorrect words captured from students' dictation notebooks at forth and fifth grade of primary school since January 2005 until March 2005.
2. The incorrect typed words gathered from type centers since October 2004 until February 2005.

Kind Of Errors	Train	Test
Complex Error	226	83
Insertion/Deletion/Substitution	2817	1293
Split Word	688	307
Transposition	96	85
Total Number Of Words	3827	1768

Table 1. Persian Misspell Word Database

There are 5595 misspelled words and their corrected forms in database. Database has been randomly divided into two parts: one for training and the other for system testing. Training set contains 3827 words and Test set has 1768 entry. Table1 shows the number and Type of errors in each part. Data set is available at http://www.digitalclone.net/localization/spellchecker. Table 2 shows examples for every type of errors in CloniZER Spell Checker Data Set with their translation.

The norm of system's precision will be calculated according to the formula (7). The presented formula for precision, is different from its classic form, regard the importance of word's rate in the suggested words' list.

$$\Pr ecision = \sum_{i}^{N} \frac{Rank_i}{10 * N} \\ Rank_i = \begin{cases} 11 - SuggestNo_{w_i} & 1 \le SuggestNo_{w_i} \le 10 \\ 0 & Otherwise \end{cases} \quad (7)$$

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

Error Type	Misspell	Correct Form	Translation
Complex	ترمسي /trmsi/	ترسیمی /trsimi/	Descriptive
Substitution	مريت /mrit/	مزيت /mzit/	Advantage
Deletion	لزامي /lzami/	الزامي alzami	Obligatory
Insertion	استهخراج asthKraj	استخراج astKraj	Extraction
Split Word	مشخصشد /mSKsSd/	مشخص شد /mSKs Sd/	is specified
Transposition	محترك /mhtrk/	متحرك /mthrk/	Moveable

Table 2. Samples of some misspell words in CloneiZER Spell Checker Data Set

The lexicon which is used contains more than 40,000 common Persian words which are extracted from Shargh[4] online newspaper from April 2004 up to September 2004.

At first, COT matrix cells are initiated by 100 and for simplicity we consider LSL set to 10 as fixed value. α ,the learning rate, is set to 0.01 according to (4). Also λ, Maximum allowed dissimilarity rate, and γ, suggest word count limitation rate, are set to 35 and 1.5.

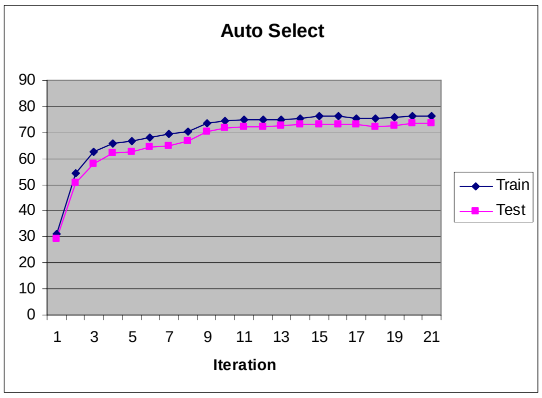

Figure 4. Auto-Select Precision

After each iteration, the system response on test set is shown in Table2 and Table3. As figure 4 and 5 shows, since the errors' sources are various, ever since iteration 15, the change in learning amount is not noticeable. In other words, error model learning, itself, results in appearing other errors in the whole database, although it can cause improvement in some other errors.

[4] www.SharghNewspaper.com

Figure 6 displays the number of words which system failed in generating any suggested word for. As previously mentioned, ever since iteration 15, suggestion learning for some of words results in system's inability in word suggestion, for some of the other words which it was previously able to provide a suggestion list for them.

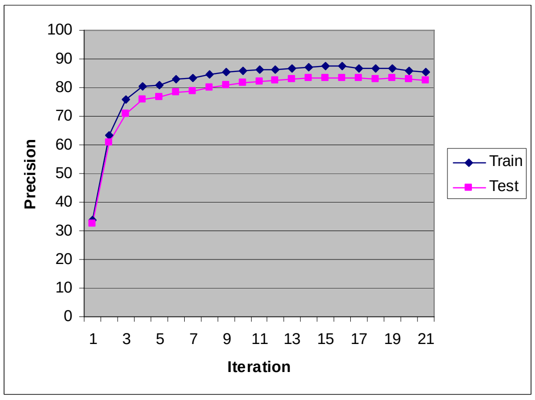

Figure 5. System Precision (Percentage/Iteration No.)

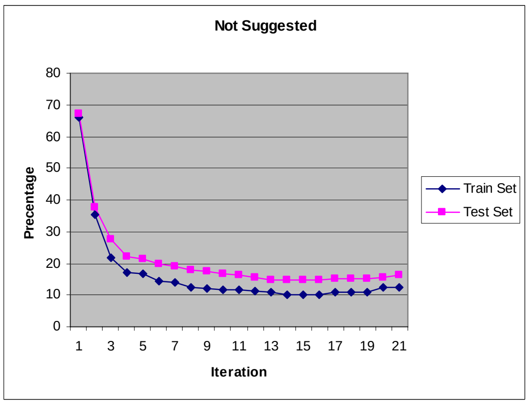

Figure 6. No Suggestion Precentage (Precentage/Iteration No.)

5. Conclusion

Popper definition of media error pattern is one of the most important issues in spell checking problem. Also, tuning or adaptation of the defined error pattern has an important role. Therefore, most of proposed spell checkers need a lot of modifications to apply in another language or application such as pre-processing.

In this paper, we introduce a novel language independent and 'built-in error pattern free' system, for spell checking; in other words we proposed a variable error pattern model which can learn the media error pattern in contrast to other algorithms. It learns error pattern with some sample from language or media. Our proposed method can adapt and tune itself by interactions by user or outer media and it improves its suggestion list as time goes by.

In CloniZER spell checker system, adding a new language is equal to adding a lexicon and some uniform sample for learning. The system will extract the common error patterns without the help of expert. Briefly, the proposed approach has more flexibility, more accuracy, data compression rate, and reliability with comparing to other proposed methods. However, our method is sensitive to media, but it shown that it has an acceptable result for auto-selection problems.

6. 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” , IASTED 2003, 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, 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.

Iteration No.	No Suggestion	Index Number in Suggested List									
		1	2	3	4	5	6	7	8	9	10
1	2526	1189	102	10							
2	1339	2053	337	48	4	6					
3	833	2387	474	75	21	8	9	9	7	1	3
4	652	2526	499	89	28	9	6	8	4	5	1
5	637	2544	500	87	21	16	5	6	3	6	2
6	554	2601	520	85	20	20	10	4	2	5	6
7	537	2651	481	86	29	17	7	4	7	6	2
8	482	2698	477	96	29	19	9	5	4	5	3
9	464	2819	383	85	27	19	13	4	5	5	3
10	449	2842	368	81	34	16	14	7	5	5	6
11	439	2861	367	75	39	12	14	8	3	8	1
12	433	2865	368	73	37	19	11	6	2	8	5
13	419	2873	356	83	39	20	15	6	4	9	3
14	389	2889	380	77	43	15	10	9	4	6	10
15	385	2915	357	88	34	21	8	8	4	6	1
16	384	2919	354	86	35	15	8	11	7	7	1
17	417	2890	351	91	32	17	7	10	5	6	1
18	416	2886	344	93	38	11	10	14	5	8	2
19	418	2896	341	94	32	16	7	10	7	5	1
20	473	2922	289	79	28	6	5	14	8	2	1
21	483	2915	294	76	20	11	10	5	3	7	3

Table 3. Train Set Result.

Iteration No.	No Suggestion	Index Number in Suggested List									
		1	2	3	4	5	6	7	8	9	10
1	1189	519	58	2							
2	669	899	177	19	3		1				
3	486	1029	208	36	1	4	1		1	1	1
4	390	1095	229	35	8	6	2	1	1	1	
5	377	1109	223	40	8	4	4	2		1	
6	348	1139	230	33	7	4	3	2		1	1
7	339	1148	220	42	8	5	2	2		2	
8	314	1181	212	38	12	3	3	3	1	1	
9	306	1244	164	34	10	2	3	3	1	1	
10	296	1272	140	35	13	4	3	2	1	1	1
11	285	1274	150	35	11	4	6	2		1	
12	273	1277	159	33	12	5	4	2	1	1	1
13	264	1285	155	36	14	4	5	3		2	
14	259	1292	155	36	12	6	3	2	1		2
15	259	1294	153	37	13	6	4	1		1	
16	259	1294	154	37	13	3	4	3	1		
17	266	1291	153	35	12	5	4	1		1	
18	268	1280	156	39	14	3	6	1	1		
19	266	1285	159	35	12	5	4	1		1	
20	274	1297	142	34	9	5	5		1		1
21	285	1299	143	24	8	1	4	2		1	1

Table 4. Test Set Result.