Instant Diacritics Restoration System for Sindhi Accent Prediction using N-Gram and Memory-Based Learning Approaches

Hidayatullah Shaikh, Javed Ahmed Mahar, Mumtaz Hussain Mahar
Department of Computer Science,
Shah Abdul Latif University,
Khairpur Mir's,
Sindh, Pakistan

***Abstract*---The script of Sindhi Language is highly complex due to many complexities including abundance of homographic words. The interpretation of the text turns so tough due to the possibility of multitudinal meanings associated with a homographic word unless given specific pronunciation with the help of diacritics. Diacritics help the readers to comprehend the text easily. Due to the rapidly developing nature of this era, people don't bother writing diacritics in routine applications of life. Besides creating difficulties for human reading, the absence of diacritics does also make the text abstruse for machine reading. Relatively alike human, machines may also lead to semantic and syntactic complexities during computational processing of the language. Instant diacritics restoration is an approach emerged from the text prediction systems. This type of diacritics restoration is an unprecedented work in the realm of natural language processing, particularly in Indo-Aryan languages. A proposition for a framework using N-Grams and Memory-Based Learning approach is made in this work. The grab-point of this mechanism is its 99.03% accuracy on the corpus of Sindhi language during the experiments. The comparative edge of instant diacritics restoration is its being source of expedition in the performance of other natural language and speech processing applications. The future development of this approach seems vivid and clear for Sindhi orthography is highly similar to those of Arabic, Urdu, Persian and other languages based on this type of script.**

Keywords*--Sindhi Language; Instant Diacritics Restoration; Text Prediction; N-Grams; Memory-Based Learning

I. INTRODUCTION

Sindhi orthography abounds in such words which possess different meaning but identical morphological structure. These words are called homographs in linguistics. The solution to this problem is the assignment of diacritic marks to the homographs. Sindhi orthography has two types of diacritic signs used for the correct pronunciation of the words [1]. The superscript signs assigned over the letters and subscript ones beneath the letters. The routine scripts of Sindhi language are written without diacritics such as newspapers, magazines and books. Such absence brings about critical challenges facing computational processing of the language [2]. In more elaborate way, homographic words can be interchangeably meant or interpreted if diacritics are absent. They may be meant and pronounced erroneously as well. Without disambiguation, it is rather difficult to figure out the intended meaning and pronunciation of words during the process of different linguistic and speech processing applications.

The automatic assignment of diacritics in Sindhi script is essential for its processing into natural language and speech applications [3] [4]. Therefore, the literature of this type of research is replete with the details of the research works on diacritic restoration particularly by using statistical approaches [5] [2]. Firstly, the results of previous research works are not satisfactory or at acceptable level and secondly, the instant diacritics restoration is taken into consideration for the first time for Sindhi. The objective of the study is the development of automatic system that will convert the un-diacritized words into the diacritized ones by assigning the diacritic signs instantly during typing.

This research study aims at the development of automatic system that assigns diacritics to the words which at first are un-diacritized during typing instantly. For this, an investigative study with the combination of N-Grams and Letter-Level Approaches is carried out to meet the objective.

The rest of the paper is organized as follows: some research contributions of diacritics restoration of Arabic script-based languages are presented in Section II. The overview of corpus preparation is given in Section III. The proposed model for the task of instant diacritics restoration is described and depicted in Section IV. In Section V, execution process of developed software application is explained, while in Section VI, implementation process of proposed model and detail evaluation of calculated results are given and finally, the paper is concluded in Section VII with core results and conclusion.

II. RELATED WORK

The study of literature on this topic reveals that diacritics restoration is performed at letter and word level. Diacritics restoration has been centered by using various techniques at word and letter level as well, like N-Grams [6] [7], Neural Networks [8], Maximum Entropy [9], Memory-Based Learning [10] [11], and Weighted Finite State [12]. Majority of researchers has received encouraging results at word level using N-Gram language model [6] [7] [2] whereas Memory-Based Learning Approach [13] also yields good results at

letter level for the same task on Arabic script-based languages including Sindhi [14].

The task of automatic Sindhi diacritics restoration is mainly considered and taken by the researchers using statistical approaches such as maximum entropy [1], N-grams [5] and memory-based learning approach [14]. The acceptable results are achieved with memory-based learning and N-gram based language modeling approaches. Hence, the proposed instant diacritics restoration mechanism is also based on the N-Grams and Memory-Based Learning approaches. Making use of this mechanism high accuracy in less time is attained.

III. CORPUS PREPARATION

As a matter of fact, two types of data sets are always required for experimentation of diacritics restoration systems [1]. Therefore, two types of corpora are designed and developed. The first subsumes complete diacritized text and the second undiacritized text. In addition to them, a lexicon is also built. The experiments of the proposed method were performed by making use of both types of data sets; corpora and lexicon.

A data set of corpus having 2, 65,257 words are built in Sindhi language for the purpose of training and testing the system. The organized information of the developed corpus in is given in Table I. The corpus is classified into three segments: the antique books that are completely written with diacritics like Shah Jo Rosalo [15], the poetry books that possess partially diacritized text and the recently published text of different genres which are entirely void of diacritics like newspapers, magazines and text books.

TABLE. I. WORDS INFORMATION OF DEVELOPED SINDHI CORPUS

Type of Corpus	No. of Sentences	No. of Words
Fully Diacritized	8326	49,462
Partially Diacritized	10190	93,188
Not-Diacritized	14869	1,22, 607
Total	**33385**	**2, 65,257**

A. Developed Lexicon

In addition to the development of Sindhi corpus, a lexicon of Sindhi text has been created for it is an essential component for the proposed method of instant diacritization. The mechanism of the instant diacritics restoration has the basis of memory based learning approach with the aid of letter level learning approach. Relatively, a table having the letters in different forms of diacritized as well as un-diacritized is developed. The specimen of this table is given in "Fig. 1". It should be noted here that each letter is assigned a unique number for the identification. This identification is required for the execution of the letters into the system.

IV. PROPOSED MODEL

The nine components work altogether as the constituents of the proposed mechanism: Calculation of word probabilities, specimens of letters, pattern matching and comparative function of homographic structures, K-NN Classifier and Class Labels, calculation of distance between instances using overlap metric, calculate the features weight, nested hash and tokenization. The proposed model in "Fig. 2" is used to show the execution process of the complete system.

The corpus functions as a patron on which the probabilities are dependent; hence, training corpus design is a delicate matter to deal with. The more specified training corpus leads to the more accurate probabilities which help the task to be achieved conveniently. The N-grams are probabilistic models that help the provision of direction for the assignment of probabilities to the words. The unigram, bigram, trigram and so on models are used for the calculation of probabilities. A unigram is an N-gram of 1, bigram of 2, and consequently trigram of 3, and so on with the progressive numbers [16]. The text is a sequential series of structured words and can be given representation as below:

$$P(W_1, W_2, \ldots W_{n-1}, W_n) \quad (1)$$

For a bigram grammar

$$P(w_1^n) \approx \prod_{i=1}^{n} P(w_i \mid w_{i-1}) \quad (2)$$

The trigram is same as bigram except the condition on two previous words as under.

$$P(w_1^n) \approx \prod_{i=1}^{n} P(w_i \mid w_{i-2} w_{i-1}) \quad (3)$$

The ultimate product on the part of the system is the provision of the option to the user to choose the suitable or correct words as per the requirement. Therefore, the language modeling is used for the computation of N-Grams up to quad one. The probabilities of all the words given in the corpus are individually calculated and stored into a specified table in the designed lexicon. The purpose of this whole process is to support the further process of the mechanism.

tblLetterLevel: ...Base\DBINSDR.sdf)

	ID	Letter	Word	LetterV	WordV
▶	100001	ا	اِنَسانَ كَ	ا	انسان كـ
	100002	نَ	نَسانَ كَيَ	ن	نسان كي
	100003		سانَ كَيَ		سان كي
	100004	كَ	انَ كَيَ سَ	ك	ان كي س
	100005	يَ	نَ كَيَ سَتيَ	ي	ن كي سي
	100006		كَيَ سَتيَ		كي سي
	100007	سَ	كَيَ سَتيَ كَ	س	كي سي ك
	100008	يَ	يَ سَتيَ كَا	ي	ي سي كا
	100009		سَتيَ كَانَ		سي كان
	100010	كَ	سَتيَ كَانَ	ك	سي كان
	100011	ا	يَ كَانَ بَ	ا	ي كان ب
	100012	ن	كَانَ بَهَ	ن	كان به
	100013		كَانَ بَهَر		كان بهر
	100014	بَ	انَ بَهَريَ	ب	ان بهري
	100015	هَ	ن بَهَرِيَنَ	ه	ن بهرين
	100016	ر	بَهَرِيَنَ	ر	بهرين
	100017	يَ	بَهَرِيَنَ بَ	ي	بهرين ب
	100018	ن	هَرِيَنَ بَنَ	ن	هرين بن
	100019		رِيَنَ بَنَهَ		رين بنه
	100020	بَ	يَنَ بَنَهَنَ	ب	ين بنهن
	100021	نَ	ن بَنَهَنَجَ	ن	ن بنهنج
	100022	هَ	بَنَهَنَجَنَ	ه	بنهنجن
	100023	نَ	بَنَهَنَجَنَ	ن	بنهنجن

Fig. 1. Sample Database Table for Instant Diacritics Restoration

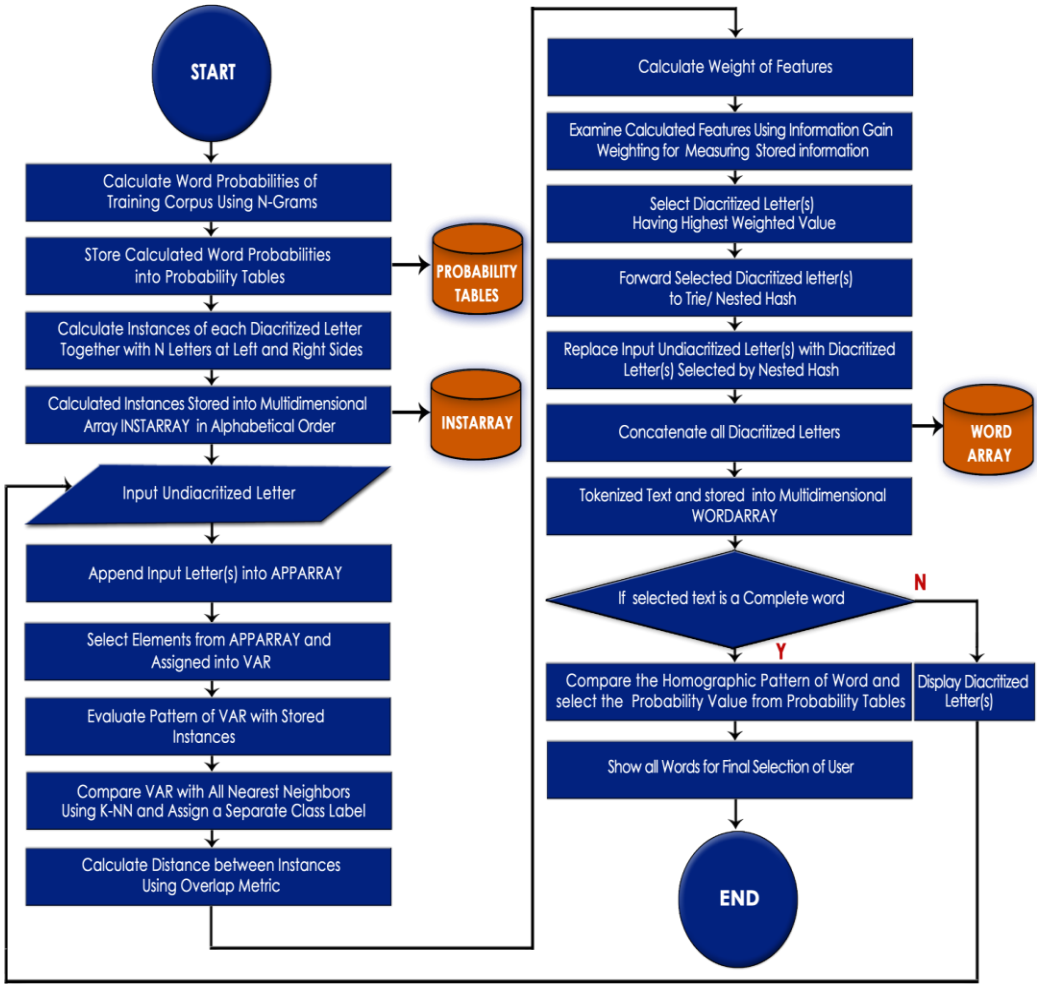

Fig. 2. Proposed Model for Sindhi Instant Diacritics Restoration

After the words probabilities are calculated, the system starts computation of the available instances of each diacritized letter. For this, almost all the possible instances of all the letters in corpora calculated with every diacritic mark; i.e., بُ , بِ ,بَ are calculated altogether with the surrounding letter (N letter) on both left and right sides. At the same time, the calculated instances are saved in a multidimensional array ascending. At least 1224688 instances are taken from the available corpus taking care of the particular notations given to the white spaces (SP), commas (CO) and dots (DO) alike [11] [13]. A vector based multidimensional array is used for the storage of these examples. The corpus same from [1] is given below and the related sample of feature vectors extracted from the same source is presented in Table II.

كِنهن بہ نِظَامَ گي سَنوَارِڻُ ۽ بِگاڙِڻُ ۾ آسَان سَينِي جو
كِٿي نہ كِٿي هَڏُ ضَرُورِ هُوندَو آهي آسَان مَان هَرِ هِكَ
مَاڻهُون پَاڻُ مَان بَدِيَائتِي كَدِي جَڏي دِيَائتدَارِي پَنهنجو
پَاڻُ پَيدَا ٿِي پَوَندِي آسَان سَيُ كُوڙَ گَان پَاسو كَيُون سَچُ
پَاڻهِي سَامهُون آجِي وِيندَو

TABLE. II. SAMPLE LETTERS AND FEATURE VECTORS

Letters	Feature Vectors
ڪ	: ا ,ن ,ت ,ي ,SP ,چ ,ڏ ,SP ,ي ,ڊ : پ ,ا ,س ,و ,SP ,SP ,CO ,ن ,و ,ي
ڪ	: ن ,هه ,ن ,SP ,ب ,SP ,SP ,SP ,SP ,SP : ي ,SP ,ڪ ,ٽ ,SP ,و ,ج ,SP ,ي : ٽ ,ي ,ن ,هه ,ٽ ,SP ,هه ,ٽ ,ي ,SP
ڪ	: SP ,م ,ا ,ڻ ,و ,هه ,SP ,ر ,هه ,SP : و ,ڙ ,SP ,ک ,ا ,SP ,پ ,س ,SP ,ن
هَ	: ٽ ,SP ,ض ,ر ,و ,SP ,ي ,ٽ ,ڪ ,SP : ي ,DO ,ا ,س ,ا ,ا ,SP ,و ,د ,ن : ر ,SP ,هه ,ڪ ,SP ,SP ,ن ,ا ,م ,SP : ر ,ي ,SP ,پ ,ن ,پ ,SP ,و ,ج ,ن
هِ	: ن ,SP ,ن ,هه ,ن ,ن ,ڪ ,SP ,SP ,SP : ڪ ,SP ,م ,ڻ ,هه ,SP ,ر ,هه ,SP ,ن : ي ,SP ,س ,ا ,م ,ڻ ,ا ,پ ,SP ,چ
هُ	: و ,ن ,د ,و ,SP ,SP ,ر ,و ,ر ,ض : و ,SP ,پ ,ا ,ڻ ,ڻ ,ا ,م ,SP ,ڪ : و ,ن ,SP ,ا ,چ ,م ,ا ,س ,SP ,ي

The absence of diacritical marks lead to many complexities in the text regarding various possible vowels sounds used in a word [11]. The word سکن may be taken for example. The system performs comparison of the pattern of the un-diacritized word with the diacritized ones available in the corpus. System receives two types of words سُکَنِ and سِکَنِ. Pattern matching process is carried out using regular expression approach. The system, then, acknowledges the pattern of un-diacritized input word with the diacritized one. The suitable word on the basis of the highest probability is fixed at the same location. Sample regular expression example is given graphical representation below:

سُکَنِ:-

Step-1 " س (و) ,ک (-) ,ن (_)"

Step-2 {س (و) ,ک (-) ,ن (_) =}

Step-3 {اکر س (و) ,اکر ک (-) ,اکر ن (_)=}

(سُ) (ک) (ن) = (سُک) , (سُن) , (کَن)

سِکَنِ:-

Step-1 " س (_) ,ک (-) ,ن (_)"

Step-2 {س (_) ,ک (-) ,ن (_) =}

Step-3 {اکر س (_) ,اکر ک (-) ,اکر ن (_)=}

(سِ) (کَ) (نِ) = (سِک) , (سِن) , (کَن)

The complete group of examples is extracted from the corpus for each complex letter structure. Each letter from the set is taken one by one including the surrounding neighbors from both sides. Then, the system compares with the available instances in the corpus. The KNN classifier is used for this comparison process. The value of each feature vector is calculated and stored in the built-in metric. All of the values of each feature are weighted and tagged with labels whether matched or mismatched structures. These instances are divided in accordance with the assigned labels. The instance based learning algorithm is taken into use for the comparison of new problem examples with instances stored already in the memory. K-nearest neighbor algorithm is the proven simplest method of an instance-based learning one; on the other hand, K-NN method categorizes the objects based on the nearest training example in the feature space. The core model is given below [17]:

$$f(x_q) = \frac{\sum_{i=1}^{k} f(x_i)}{k} \quad (4)$$

All of the input instances are compared individually with the all the closest neighbors by using KNN classifier. Finally, the system accepts the most frequent ones. A multidimensional array in the system saves the training examples containing feature vectors. The label specifies each example according to its class. The highest numbers of votes including with neighbors categorize the labeled entity.

While the process of classification undergoes, a unique test instance is fed to the system, using the distance Δ(X, Y). This computes the sameness of the new examples and all of the other examples in memory. Overlap metric is used for this task particularly considering the distance between instances manifested by N-features. It is only to show the distance per feature [13] [14].

$$\Delta(X, Y) = \sum_{i=1}^{n} \delta(x_i, y_i) \quad (5)$$

The metric performs counting of the entire number of feature-values in both patterns regardless of matching or mismatching for the addition of the domain knowledge bias to the weight.

For the weight of the features, statistical information is calculated through an examination to reach the better predictors of the class tags. Information Gain (IG) examines each feature individually and prepares measurement for the information to be produced and stored knowledge for valid class label.

Immediately after the above process, hash table begins the process of storing data in an associated network manner. This table stores the data in the array format and each data value receives a unique index within. This way the data is quickly accessed after knowing the index of the required data. Hashing technique is widely known technique that is used for the conversion of a range of key values to a range of the array indexes.

Tokenization of the script of Sindhi is also one of the challenging tasks due to the complexities in the text,

particularly the complexities of homographic structures. A compound word needs to be entitled as a single token but the embedded space required in between creates ambiguity for the tokenization process. The embedded space is required in between due to the cursive nature of Sindhi script and its connecting and non-connecting letters. Therefore, more attention is to be paid because of these complications facing the tokenization. Mahar's [1] tokenization model is taken in this research project.

In fact, Sindhi script abounds in homographic words. As a result, the ambiguity is often observed when the text is undiacritized. A simple word and root word of Sindhi قسم has such constituent letters which may be interchangeably taken in almost two way as قَسَمُ (an oath) (noun), قِسِمُ (kind) (noun). The taken words without diacritics are exactly identical. Thus, they create ambiguity for NLP applications. Viterbia Algorithm is one of the efficient approaches to find the most likely path transitions in such cases. This algorithm produces the most likely possible word on the basis of the highest probability value calculated by using N-grams [16].

V. Execution Process of Application

Text prediction is the basic idea that ignition to the Instant Diacritics Restoration. The former was proposed to save time and energy simultaneously by offering assumptions of possible upcoming set of letters after typing the beginning letters of words. By typing each succeeding letter, the user receives possible suggestions in different forms of popup to adopt with a single click only rather than typing all the upcoming letters of the word. For example, user wants to type the word انسان.

After typing the first letter, he will be shown some popup carrying some most possible and frequently used words begging with ا. Then, he will type the next letter ن, he will again be shown some set of most possible and frequently used set of letters after the two begging ones. If he finds the same letter in the popup, he would just hit a single click to get the word typed rather than hitting five strokes for all the five letters in the word.

This function of text prediction gave birth to the idea of instant diacritics restoration. The predictive approach of instant diacritization facilitates the user to type the words with their exact pronunciations which further helps in reading it correctly. The editor actively and simultaneously works with the user and assigns the diacritics automatically. The user has to type the words only. The diacritics will automatically be assigned immediately. For example, the user wants to type the word اِنْسَانُ, he first types the first letter ا, the editor will assign it the superscript diacritic sign ـَ initially, for the system is assigned this task for every first letter. After ا, the user types another letter ن, the system will immediately calculate the probability of the possible diacritics to this couple of letters and assign ـْـ to ن, simultaneously the ـَ to ا will change into ـٰ .

The user is to type س now, as he types س the system again goes for the calculation of the probability of the possible diacritics to this combination of letters and assigns the diacritics to all of the three according the highest found match in the corpus. Now, the user moves ahead to type ا and then ن, the system will simultaneously work with the letters and the diacritics while calculating the probabilities of the letters and diacritic signs from the given corpus. After the user is done with typing اِنْسَانُ, the system finalizes its diacritics with the same procedures detailed above. The same process takes place by typing each letter in the editor.

VI. Implementation And Results

The training and testing set design stand as the foundations to the final results. Therefore, both are mainly concerned till the results are derived. Different techniques like Word Error Rat, Diacritic Error Rate, Precision, Recall and F-measures were in the use previously. We have also taken Precision which is one of them due to the fact that its performance is observed to be better at letter level approach [1]. Moreover, the complex letters assign the target features for being trained; hence, the task is performed at the lowest basic level of letters. Three mainly used diacritics, i.e., Zabar, Zair and Pesho in Sindhi are considered in experiments.

The Letter Level Learning method processes every letter taken from the corpus and creates a ten letters vector. Each vector is put into an array. Consequently, each letter is pre-processed with its calculated probability. After receiving the testing data set, system throbs the comparison of all the undiacritized letters of the testing data set with the pre-processed data available in the arrays and after the said process replace the letter with the diacritized one.

From the total sets of instances taken from the developed corpus, 159330 instances are experimentally tested from each set. The testing examples are approximately 15% of the whole set of examples. Table III, Table IV and V depict the results attained with N=1, 3 and 5. The tables show the ambiguous letters extracted from the developed corpus, the precision as the result by applying instance-based learning at letter level.

TABLE. III. Ambiguous Set of Letters, Examples and Achieved Precision with N=1

Ambiguous Set	Total Examples	Tested Examples	Precision Achieved
اَ أَ اِ	99,262	14889	91.22%
بَ بُ بِ	15,881	2383	93.51%
بْ بٌ بٍ	6,447	967	92.71%
پَ پُ پِ	14,752	2212	90.84%
تَ تُ تِ	34,169	5126	91.36%
تّ تٌ تٍ	11,223	1684	90.33%
تْ تَ تً	10,227	1534	92.42%
ٹَ ٹُ ٹِ	4,673	701	90.01%
ٹْ ٹً ٹٍ	850	127	89.19%
ٹّ ٹَ ٹُ	12,273	1841	92.62%
جَ جُ جِ	41,688	6253	88.24%
جْ جً جٍ	5,486	823	83.61%
چَ چُ چِ	782	117	94.56%
چْ چً چٍ	238	36	94.62%
چّ چَ چُ	18,852	2828	90.41%
چِ چْ چً	10,293	1544	92.55%
حَ حُ حِ	20,790	3118	93.77%
خَ خُ خِ	8,039	1206	95.71%

دُ دِ دَ	30,477	4572	97.09%
دٌ دٍ دً	993	149	94.22%
بْدِ دِ دِ	274	41	95.11%
دَّ دُّ دْ	25,622	3843	94.63%
بْدِ دِ دِ	691	104	96.12%
دَ دِ دَ	532	80	90.81%
رُ رِ رَ	48,033	7205	90.01%
رَّ رُّ رِّ	1,943	291	93.32%
زُ زِ زَ	849	127	90.54%
سُ سِ سَ	24,237	3635	94.90%
شُ شِ شَ	994	149	94.32%
صُ صِ صَ	592	89	94.88%
ضُ ضِ ضَ	231	35	95.62%
طَ طِ طَ	838	126	89.21%
ظَ ظِ ظَ	201	30	90.01%
عُ عِ عَ	11,421	1713	93.79%
غُ غِ غَ	841	126	93.88%
فُ فِ فَ	12,840	1926	94.56%
قُّ قِّ قَّ	556	83	93.76%
قُ قِ قَ	605	91	94.55%
كُ كِ كَ	54,837	8226	95.64%
كُ كِ كَ	28,444	4267	95.99%
گُ گِ گَ	14,766	2215	94.06%
گھ گھ گھ	2,495	374	81.58%
گُ گِ گِ	348	52	94.93%
گُّ گِّ گَّ	173	26	92.27%
لُ لِ لَ	55,121	8268	92.77%
مُ مِ مَ	60,270	9041	95.74%
نُ نِ نَ	101,126	15169	90.31%
نُّ نِّ نَّ	126	19	90.91%
وُ وِ وَ	55,664	8350	95.05%
هِ هَ هه	84,033	12605	88.64%
ءَ ءُ ءِ	76	11	93.03%
يُ يِ يَ	126,023	18904	90.88%

TABLE. IV. AMBIGUOUS SET OF LETTERS, EXAMPLES AND ACHIEVED PRECISION WITH N=3

Ambiguous Set	Total Examples	Tested Examples	Precision Achieved
اَ اُ اِ	99,262	14889	94.55%
بَ بُ بِ	15,881	2383	96.86%
بَ بُ بِ	6,447	967	94.66%
پُ پِ پَ	14,752	2212	95.14%
تَ تُ تِ	34,169	5126	96.31%
ٹُ ٹِ ٹَ	11,223	1684	92.23%
ٹُ ٹِ ٹَ	10,227	1534	93.76%
ثُ ثِ ثَ	4,673	701	95.85%
ثُ ثُ ثِ	850	127	94.63%
پَ پُ پِ	12,273	1841	92.62%
جَ جُ جِ	41,688	6253	92.54%
جَهه جِهه جُهه	5,486	823	87.41%
چُ چِ چَ	782	117	95.33%
چُ چِ چَ	238	36	97.02%
چُ چِ چَ	18,852	2828	94.48%
چُ چِ چَ	10,293	1544	95.88%
حُ حِ حَ	20,790	3118	96.77%
خُ خِ خَ	8,039	1206	96.07%
دُ دِ دَ	30,477	4572	98.21%
دٌ دٍ دً	993	149	95.99%
بْدِ دِ بَ	274	41	96.79%
دَّ دُّ دْ	25,622	3843	97.13%

بْدِ دِ بَ	691	104	96.88%
دَ دِ دَ	532	80	93.22%
رُ رِ رَ	48,033	7205	93.66%
رَّ رُّ رِّ	1,943	291	96.22%
زُ زِ زَ	849	127	94.34%
سُ سِ سَ	24,237	3635	95.42%
شُ شِ شَ	994	149	97.32%
صُ صِ صَ	592	89	95.07%
ضُ ضِ ضَ	231	35	97.65%
طَ طِ طَ	838	126	93.44%
ظَ ظِ ظَ	201	30	93.71%
غُ عِ عَ	11,421	1713	95.17%
غُ غِ غَ	841	126	95.48%
فُ فِ فَ	12,840	1926	95.06%
قُّ قِّ قَّ	556	83	96.72%
قُ قِ قَ	605	91	95.15%
كُ كِ كَ	54,837	8226	96.99%
كُ كِ كَ	28,444	4267	97.01%
گُ گِ گَ	14,766	2215	95.06%
گھ گھ گھ	2,495	374	87.25%
گُ گِ گِ	348	52	95.91%
گُّ گِّ گَّ	173	26	94.87%
لُ لِ لَ	55,121	8268	96.44%
مُ مِ مَ	60,270	9041	97.14%
نُ نِ نَ	101,126	15169	96.53%
نُّ نِّ نَّ	126	19	95.11%
وُ وِ وَ	55,664	8350	96.57%
هِ هَ هه	84,033	12605	91.84%
ءَ ءُ ءِ	76	11	93.78%
يُ يِ يَ	126,023	18904	96.77%

TABLE. V. AMBIGUOUS SET OF LETTERS, EXAMPLES AND ACHIEVED PRECISION WITH N=5

Ambiguous Set	Total Examples	Tested Examples	Precision Achieved
اَ اُ اِ	99,262	14889	98.26%
بَ بُ بِ	15,881	2383	99.17%
بَ بُ بِ	6,447	967	99.09%
پُ پِ پَ	14,752	2212	99.74%
تَ تُ تِ	34,169	5126	99.22%
ٹُ ٹِ ٹَ	11,223	1684	99.04%
ٹُ ٹِ ٹَ	10,227	1534	98.51%
ثُ ثِ ثَ	4,673	701	99.64%
ثُ ثُ ثِ	850	127	99.61%
پَ پُ پِ	12,273	1841	99.55%
جَ جُ جِ	41,688	6253	98.14%
جَهه جِهه جُهه	5,486	823	94.38%
چُ چِ چَ	782	117	99.23%
چُ چِ چَ	238	36	99.88%
چُ چِ چَ	18,852	2828	99.66%
چُ چِ چَ	10,293	1544	99.17%
حُ حِ حَ	20,790	3118	99.47%
خُ خِ خَ	8,039	1206	99.47%
دُ دِ دَ	30,477	4572	99.91%
دٌ دٍ دً	993	149	99.87%
بْدِ دِ بَ	274	41	99.73%
دَّ دُّ دْ	25,622	3843	99.44%
بْدِ دِ بَ	691	104	99.81%
دَ دِ دَ	532	80	99.88%
رُ رِ رَ	48,033	7205	99.22%
رَّ رُّ رِّ	1,943	291	99.11%

زْ ز ز	849	127	99.14%
سْ س س	24,237	3635	98.66%
شْ ش ش	994	149	98.93%
صْ ص ص	592	89	99.28%
ضْ ض ض	231	35	99.33%
طْ ط ط	838	126	99.17%
ظْ ظ ظ	201	30	99.32%
عْ ع ع	11,421	1713	99.37%
غْ غ غ	841	126	99.57%
فْ ف ف	12,840	1926	99.22%
قْ ق ق	556	83	99.13%
قْ ق ق	605	91	97.55%
كْ ک ک	54,837	8226	99.18%
كْ ک ک	28,444	4267	99.63%
گْ گ گ	14,766	2215	99.26%
گھْ گھ گھ	2,495	374	94.52%
گْ گ گ	348	52	99.01%
گْ گ گ	173	26	99.61%
لْ ل ل	55,121	8268	99.14%
مْ م م	60,270	9041	99.93%
نْ ن ن	101,126	15169	99.44%
ںْ ں ں	126	19	98.66%
وْ و و	55,664	8350	99.51%
هْ ھ ه	84,033	12605	97.35%
ءْ ء ء	76	11	98.17%
یْ ي ي	126,023	18904	99.26%

Three different window sizes were tested to reach the best one. Among the window sizes of two, six, and ten letters (i.e., N= 1, 3, 5), the calculated accuracy with N=1 is 92.52%, accuracy of 95.12% is received when N=3 and 99.03% is calculated with N=5. Window size for the greatest and most efficient accuracy was observed up to ten nearest accompanying letters (i.e., N=5) where N stands for the number of letters from each side of the letter under process. The calculated cumulative precisions with different experimented window sizes are shown in "Fig.3".

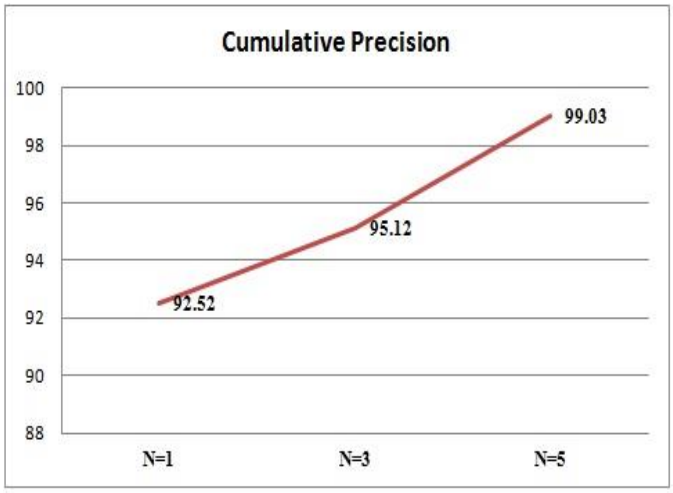

Fig. 3. Calculated Cumulative Precision with Different Window Sizes

The figures, given in the tables, show that a considerable difference can be found among them; in addition to this, the calculated results reveal that the window size is also decisive in increase and decrease of results. Therefore, N=5 proves to be the most suitable and reliable window comparatively.

VII. Conclusion

Automatic instant diacritic restoration is essential component for many NLP applications. The restoration is attempted with the most possible intelligent use of two approaches; N-grams based and Letter Level Learning-based. Each of both methods has their own specifications along with the limitations. The proposed mechanism in this study is experimented on our developed corpus of Sindhi language. The window (N=5) is found the best one after testing different sizes. The Precision with this window is achieved at 99.03%. The proposed method is also capable for the instant diacritics restoration of Arabic, Urdu and Persian languages after slight modifications.

References

[1] J. A. Mahar, "Statistical Approaches to Diacritics Restoration in Sindhi Text to Speech Synthesis System", PhD Thesis, Hamdard University, Karachi, Pakistan, 2012.

[2] S. A. Mahar, "Comparative Analysis of Vowel Restoration for Arabic Script Based Languages Using N-Gram Models", MS Thesis, Shah Abdul Latif University, Khairpur, Pakistan, 2014.

[3] A. Al-Wabil, H. Al-Khalifa, W. Al-Saleh, "Arabic Text-To-Speech Synthesis: A Preliminary Evaluation", In Proceedings of the 2007 World Conference on Educational Multimedia, Hypermedia and Telecommunications, Vancouver, Canada, Pp. 4423-4430, 2007.

[4] A. A. Shah, A. W. Ansari, L. Das, "Bi-Lingual Text to Speech Synthesis System for Urdu and Sindhi", National Conference on Emerging Technology, Pp. 126-130, 2004.

[5] J. A. Mahar, G. Q. Memon, "Automatic Diacritics Restoration for Sindhi", Sindh University Research Journal (Science Series), Vol. 43, No. 1, Pp. 43-50, June 2011.

[6] Y. Gal, "An HMM Approach to Vowel Restoration in Arabic and Hebrew", ACL-02 Workshop on Computational Approaches to Semitic Languages, Association for Computational Linguistic, Philadelphia, Pennsylvania, Pp.1-7, 2002.

[7] A. A. Harby, M. A. Shehawey, R. S. Barogy, "A Statistical Approach for Quran Vowel Restoration", ICGST International Journal on Artificial Intelligence and Machine Learning, Vol. 8, No. 3, Pp. 9-16, 2008.

[8] H. Sultan, "Automatic Arabic Diacritization using Neural Network", Scientific Bulletin of Faculty of Engineering Ain-Shams University: Electrical Engineering, Vol. 36, No. 4, Pp.501-510, 2001.

[9] I. Zitouni, R. Sarikaya, "Arabic Diacritic Restoration Based on Maximum Entropy Models", Computer Speech and Language, Vol. 23, Pp. 257-276, 2008.

[10] R. Mihalcea, V. Nastase, "Letter Level Learning for Language Independent Diacritics Restoration", Proceedings of 6th Workshop on Computational Language Learning, Vol. 20, Pp.1-7, 2002.

[11] S. Kubler, E. Mohamed, "Memory-based vocalization of Arabic", In Proceedings of the LREC Workshop on HLT and NLP within the Arabic World, Pp. 58-62, Morroco, 2008.

[12] R. Nelken, S. M. Shieber, "Arabic Diacritization using Weighted Finite-State Transducers", ACL Workshop on Computational Approaches to Semitic Languages, Association for Computational Linguistic, Pp.79-86, Michigan, 2005.

[13] R. F. Mihalcea, "Diacritic Restoration: Learning from Letters Versus Learning from Words", Lecture Notes in Computer Science, Vol. 2276, Pp. 96-113, 2002.

[14] J. A. Mahar, G. Q. Memon, H. Shaikh, "Sindhi Diacritics Restoration By Letter Level Learning Approach", Sindh University Research Journal (Science Series), Vol. 43, No. 2, Pp. 119-126, December 2011.

[15] K. Aadvani, "Shah Jo Risalo", 2nd Edition, Sindhica Academy, Karachi, Pakistan, 2009.

[16] D. Jurafsky, J. H. Martin, “Speech and Language Processing: An Introduction to Natural Language Processing”, Computational Linguistic and Speech Recognition, Prentice-Hall, Pp. 300-307, 2000.

[17] Y. Hifny, “Restoration of Arabic Diacritics Using Dynamic Programming," COLING, 2012.

[18] C. Lee, G. G. Lee, “Information Gain and Divergence-Based Feature Selection for Machine Learning-Based Text Categorization”, An International Journal of Information Processing and Management, Special Issue: Formal Methods for Information Retrieval, Vol. 42, Issue 1, Pp. 155-165, January 2006.