N-GRAMS SOLUTION FOR ERROR DETECTION AND CORRECTION IN HINDI LANGUAGE

Shailza Kanwar, Manoj Sachan and Gurpreet Singh
Dept. of CSE, SLIET Longowal
Sangrur, India

Abstract: Hindi is the National language of India, which is still in its early stage of research and development regarding natural language processing applications in comparison to other languages like English, Chinese. Natural language processing is a field of Artificial Intelligence, which includes major tasks such as information retrieval, word segmentation, speech recognition, parsing, part of speech tagging, text classification, automatic text summarization etc. Spelling detection and correction in Hindi language is an important task of NLP which has not gotten sufficient attention till date. Spelling detection and correction for Indian languages such as Hindi is considered as a difficult task. Hindi Language is very different from English language in its phonetic properties and grammatical rules. Thus the existing techniques and methods that are being used to check the errors in English language can’t be used for Hindi Language. There are mainly two types of error: Non word error and real word error. Error detection for non-word error in Hindi language has been done but for real word error no work has been done till date. This paper focused on Real word spelling error detection and correction in Hindi text by using N Grams Model and Levensthein edit distance algorithm.

Keywords: Natural Language Processing, Real Word Error, Spell Checking, N Grams

I. INTRODUCTION

Language processing is a very complex area of research. Automatic processing on language contents by machines directly has been considered as one of the most difficult tasks [1,2]. Till the time a lots of research work has been done on English and other languages and still more research work is going on for these languages to make them more user friendly, however very less work has been done for Hindi language to making it simpler, easier, corrective, understandable and computerized language. So it becomes more and more important to make a Hindi language user friendly as the government is emphasizing to increase its official use in letters and notifications. So this is my first step to making it more user friendly. The Spelling detection and correction techniques have an important role in various applications such as search engines, text editors and NLP tools like OCR, machine translations, text authoring systems etc. Now a days, there are several word processing applications are in use, so the solutions for spelling error correction become more important to provide accurate and quality information through text [12]. There are lots of applications and research available for English spelling detection and correction, but for Hindi very less work has been done.

English language has 21 consonants and 5 vowels, but Hindi language has 21 consonants, 10 vowels, 10 vowel sign (modifier), half characters and halant etc. So the methods applied to English language can’t be directly applied to Hindi Language. Hindi language has an ambiguity due to presence of matras and many other symbols which confuses the user in writing the correct word for example (ए, ए) (ए, ए) [13-14]. Both the words in pair looks similar, but have different meaning. User sometimes mistype the character and the resultant word leads to an error.

Spelling errors are broadly classified in two main categories: Non word Error and Real word error. Non word errors are those errors which do not exist in the dictionary. These errors have no meaning and can be detected easily by the spell checker. Real word occurs when a user mistakenly type the correct spelt word when the other word intended. These errors are difficult to find out as they exist in the dictionary, but incorrect according to the context. In this paper, author focused on correction and detection of real word errors in Hindi text.

This paper is organized as follow: Section 2 covers a brief overview of the related work. We describe our proposed method in Section 4. Evaluation and experimental results are discussed in Section 5. We conclude in Section 6.

II. RELATED WORK

Several methods have been proposed to handle real word spelling error problem. They are mainly based on either semantic information or machine learning and statistical

Based on spelling error detection and correction, a lot of work is available for English language, but for Hindi language not much work is available.

III. OVERVIEW OF HINDI LANGUAGE

Hindi is the National language of India which is spoken in different parts of India. Hindi is written in Devanagari script. Some of the major features of Devanagari script are:

- Type of writing system: alphasyllabary/ abugida, is a segmental writing system in which consonant–vowel sequences are written as a unit: each unit is based on a consonant letter, and vowel notation is secondary.

- Direction of writing: left to right in horizontal lines.

- Consonant letters carry an inherent vowel which can be altered or muted by means of diacritics or matra.

- Vowels can be written as independent letters, or by using a variety of diacritical marks which are written above, below, before or after the consonant they belong to. This feature is common to most of the alphabets of South and South East Asia.

- When consonants occur together in clusters, special conjunct letters are used.

- The order of the letters is based on articulatory phonetics.

- There are total 47 primary characters Devanagari script, out of which 14 characters are vowels and 33 letters are consonants

IV. PROPOSED METHOD

Our proposed method initially tokenizes the user input sentence into words and then creates confusion set for each candidate word by using Levenshtein distance equals to two from the corpus. Then it calculates the probability of the elements of the confusion set by using bigrams and trigrams corpus. On basis of this error is detected and words are suggested. The detailed description of the proposed method is presented below.

<table>
<thead>
<tr>
<th>Vowels</th>
<th>Consonants</th>
</tr>
</thead>
<tbody>
<tr>
<td>अ आ इ ई उ ऊ ए ऐ ओ औ</td>
<td>क, ख, ग, घ, ज, झ, छ, ज्ञ</td>
</tr>
<tr>
<td>ट, ठ, ड, ढ, ण, त, थ, द, ध, न</td>
<td></td>
</tr>
<tr>
<td>प, फ, भ, ब, म, य, र, ल, व, श, ष</td>
<td></td>
</tr>
<tr>
<td>ह, ङ</td>
<td>म, द, त, ण, त्र, त्व, द्व, र, ल, व, श, ष, ह, ङ</td>
</tr>
</tbody>
</table>

**Tokenization**

Tokenization is the process to break the block of text into a list of words. The text is broken with help of some boundary delimiter and blank space. The boundary delimiters are several punctuation marks. The Boundary delimiter could differ from font to font. In ASCII font several punctuation marks are part of the text but it is not case in UNICODE encoding system. But there are certain punctuation delimiters which are to be ignored. Of one them and the most important one is hyphen (-). Some words in the dictionary contains hyphen. Thus, hyphen can’t be used as a boundary delimiter

**Creation of Confusion Set**

A Confusion set for the main word $E^i$ is a set of words ($E_{i1}, E_{i2}, E_{i3}, ..., E_{in}$) which contains words that are likely to be confused with the main word [15]. These words are convertible to the main word with single edit operation. We use Levenshtein Distance, which is the minimum number of edit operations required to convert one word into another. An edit operation is either asubstitution, deletion or aninsertionof a character in the word. The confusion set is represented as

$$S(E^i) = \{E_{i1}, E_{i2}, E_{i3}, ..., E_{in}\}$$

where $E^i$ is the i-th word in the test sentence and $n_i$ is number of elements in the set $S(E^i)$.

**Estimation of N-Gram Probabilities**

The Probability of the sentence in bigram model is calculated by using Hidden Markov Chain rule. According to Markov rule, probability of an upcoming event (next word) depends upon prior events (previous words).[7] For example in a bigram language model for a sentence of m words $E_1, E_2, ..., E_m$ it can be calculated as

$$P(E_1, E_2, ..., E_m) = P(E_1|b)P(E_2|E_1)P(E_3|E_2)...P(E_m|E_{m-1})P(b|E^n)$$

where $b$ denotes blank.
In our model, probability of sentence is not calculated. We are taking the hidden Markov model assumption that the appearance of any event (word) depends upon its next and previous words only and independent of any other event. [7]. By using Maximum Likelihood Estimation, we get the left, right bigrams and trigrams probabilities as follows

\[
P_1(E_j | E_i^{-1}) = \frac{\text{count}(E_i^{i}, E_j^{i})}{\sum_{i=0}^{N} \text{count}(E_i^{i}, E_j^{i})} \quad (1)
\]

\[
P_2(E_j | E_i^{+1}) = \frac{\text{count}(E_i^{i+1}, E_j^{i})}{\sum_{i=0}^{N} \text{count}(E_i^{i}, E_j^{i+1})} \quad (2)
\]

\[
P_3(E_j | E_i^{-1}, E_i^{+1}) = \frac{\text{count}(E_i^{i-1}, E_i^{i}, E_j^{i+1})}{\sum_{i=0}^{N} \text{count}(E_i^{i-1}, E_i^{i+1})} \quad (3)
\]

By equation (1), we calculate \( P_1 \) of each element of confusion set for each word using left bigram count. The denominator here represents the summation of all bigrams consisting of the previous word and one word from the confusion set. The frequency of words is given in Bigrams corpus. Similarly, we compute \( P_2 \) of each element using right bigram count in the corpus by equation (2). We compute it for every element in respective confusion set so that the following condition is satisfied:

\[
\sum_{i=0}^{N} P_1(E_j | E_i^{-1}) = 1
\]

\[
\sum_{i=0}^{N} P_2(E_j | E_i^{+1}) = 1
\]

\[
\sum_{i=0}^{N} P_3(E_j | E_i^{-1}, E_i^{+1}) = 1
\]

We combine the probability estimates of equations (1), (2) and (3) into a score of evidence that a may be correct alternative to . The score can be obtained by simple addition as follows. The values obtained from equation (1) and (2) can be combined to get the final score by adding up.

\[
\text{Score}(E_i) = P_1(E_j | E_i^{-1}) + P_2(E_j | E_i^{+1}) + P_3(E_j | E_i^{-1}, E_i^{+1}) \quad (3)
\]

**Error Detection**

Consider a sentence as an example "मेरा एक सफाई कार्य है।"

Here word ‘एक सफाई कार्य’ is a real word error. The correct word against ‘एक सफाई कार्य’ is ‘एक सफाई कार्य’. The word ‘एक सफाई कार्य’ means extent and the word ‘एक सफाई कार्य’ means result. While processing the word ‘एक सफाई कार्य’, we have the confusion set {एक सफाई कार्य, एक सफाई कार्य, एक सफाई कार्य}. The probability score of each word with rank and frequency is shown in Table 2.

<table>
<thead>
<tr>
<th>Rank</th>
<th>Confusion Set Word</th>
<th>Frequency</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>एक सफाई कार्य</td>
<td>9</td>
<td>0.52</td>
</tr>
<tr>
<td>2</td>
<td>एक सफाई कार्य</td>
<td>6</td>
<td>0.35</td>
</tr>
<tr>
<td>3</td>
<td>एक सफाई कार्य</td>
<td>2</td>
<td>0.11</td>
</tr>
</tbody>
</table>

**V. RESULTS AND DISCUSSIONS**

In order to assess our method, we collected test data which contain approximately 1000 words. There are total 150 real word errors present in the test data. The real-word error in our test data are generated by single operation like substitution, deletion or insertion. People make single position character mistake during typing which becomes real-word error. The performance of our approach can be examined by two metrics: Precision and Recall. These measures are defined as

\[
\text{Precision} = \frac{\text{Number of errors correctly detected}}{\text{Total Number of errors detected}}
\]

\[
\text{Recall} = \frac{\text{Number of errors correctly detected}}{\text{Number of errors detected}}
\]
Precision is a measure of the exactness of the spellchecker’s responses and recall is a measure of the completeness of the spellchecker. The number errors not detected by the system can be calculated as:

\[
\text{Non Detected errors} = \text{Total Number of errors} - \text{Number of errors correctly detected}
\]

Hence Percent of non-detected errors = \(\frac{\text{Non-detected errors}}{\text{Total Number of words}} \times 100\%\)

Table 3 shows Precision, Recall and percentage of erroneous detection:

<table>
<thead>
<tr>
<th>Detected Real Word Error</th>
<th>Non Detected Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision</td>
<td>Recall</td>
</tr>
<tr>
<td>88%</td>
<td>90%</td>
</tr>
<tr>
<td>1%-3%</td>
<td></td>
</tr>
</tbody>
</table>

VI. CONCLUSION

Since no work has been done for detecting and correcting real word errors in hindi language so far, the results found shows that this method of detecting and correcting real word errors proved to be very useful for hindi language. This method will make the hindi language as the world class language, which helps to use the hindi language in computer world like presently used English language. We have used bigrams and trigrams data set to detect and correct the errors which makes our proposed method to achieve a precision of 88% and recall of 90%. The bigrams and trigrams data set used in this method are not vast, hence many valid bigrams and trigrams are not present are in it.

VII. REFERENCES


