Word Recommendation System for Devanagari Script

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering

in

Software Engineering

Submitted By

Premnarayan Yadav

(Roll No. 801331020)

Under the supervision of

Mr. Karun Verma

Assistant Professor

(CSED)

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

THAPAR UNIVERSITY

PATIALA – 147004

June 2015

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, *"Word Recommendation System for Devanagari Script "*,in partial fulfillment of the requirements for the award of degree of Master of Engineering in *Software Engineering* submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Mr. Karun Verma* and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

(Premnarayan Yadav)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

Mr. Karun Verma

Assistant Professor

CSED, Department

Countersigned by

(Dr. Deepak Garg)

Head

Computer Science and Engineering Department

Thapar University

Patiala

(Dr. S. S. Bhatia)

Dean (Academic Affairs)

Thapar University

Patiala

i

Acknowledgement

First of all, I am extremely thankful to my respective guide ***Mr. Karun Verma***. Assistant professor, CSED, Thapar University for his valuable guidance, advice, motivation, encouragement, moral support, sincere effort and positive attitude with which he solved my queries and provide delightful ambiance for learning, exploring and making this thesis possible. It has been a great pleaser and experience to work under his sanctuary.

I am also heartily thankful to ***Dr. Deepak Garg***, Associate Professor and Head, CSED for motivation. He sets high standards for his students and he encourages and guides them to meet those standards.

I would like to thank my family members and my friends who are dearest and precious to me for their love, encouragement, blessings and support in all respects. Most importantly, none of this would have been possible without the love and patience of my family. To my brother and friends for showing me right direction. They are constant source of love, concern, support and strength for me all these years.

Finally I would like to thank management of Thapar University for proving a great platform for learning, not just for academics but also for sports and many other creative things.

ABSTRACT

Word recommendation system (WRS) plays important role in many areas of today's digital world. Search engine(like Google, Bing *etc*.) makes great use of word prediction by predicting accurate words or sentence, by processing initial letters of a word. A word recommender system for Indian language is a necessity to promote usage of Indian languages in support of Digital India initiative. We have used a statistical based approach to predict relevant words for user input. The system also adapts itself to improve the accuracy of recommendation for that particular user by updating its database for that user.

Table of Contents

Certificate i

Acknowledgement ii

Abstract iii

Table of Contents iv

List of Figures vi

List of Tabels vii

1. INTRODUCTION 1

1.1 Language Model for Word Recommendation System 2

1.1.1 Statistical language model: 2

1.1.2 Structural language model: 3

1.2 Language Modeling terms 3

1.2.1 Linear interpolation: 3

1.2.2 Smoothing: 4

1.2.3 Discounting method: 4

1.3 Techniques Used in WRS 6

1.3.1 Chain rule of probability: 6

1.3.2 N-gram approximation: 6

1.3.3 Levenshtein distance: 7

1.3.4 Basic Introduction of WRS: 8

2 LITERATURE SURVEY 10

2.1 Some Features of POS Tagging 14

2.2 Issues in Language Model of Hindi 16

2.2.1 Large character set: 16

2.2.2 Input sequence: 16

2.2.3 Normalization: 16

2.2.4 Typographical variants: 17

2.2.5 Phonetic similarity: 17

3 PROBLEM STATEMENT 18

4 IMPLEMENTATION 19

4.1 Design of Language Model 21

4.2 Recommendation Methodology 22

4.2.1 Normalization of input: 24
4.2.2 Word processing module: 26
4.2.3 Part of Speech tags Module: 27
5 RESULTS 29
5.1 Data Results 29
5.1.1 Comparison of memory consumption: 30
5.1.2 Comparison of searching time: 30
5.2 Experimental Results 31
5.2.1 Comparison of different models with WRS: 31
6 CONCLUSION AND FUTURE SCOPE 34
6.1 Conclusion 34
6.2 Future Scope 34
REFERENCES 35
LIST OF PUBLICATIONS 38
VIDEO LINK 39

List of Figures

Figure 1.Showing Basic model of Proposed Model 9
Figure 2. Comparison of memory consumption 11
Figure 3. Comparison of search time 13
Figure 4. User Interface for WRS 19
Figure 5. Flow chart of proposed model 23
Figure 6. Virtual Keyboard of WRS 29
Figure 7. Comparison of memory consumption of Data structures for WRS 30
Figure 8. Comparison of searching time of data structure for WRS 31
Figure 9. Working of WRS model 32

List of Tables

Table 2. Sparsity Problem with bigram matrix 3
Table 3. Bigram count and probability withunseen sequence 5
Table 4 Count of unseen words after applying absolute discounting method 5
Table 5. Error correction Example using N-gram 7
Table 6. Levenshtein distance for two strings 8
Table 7. Accuracy of POS tag classes 15
Table 8. Words with similar Unicode values having different representations 17

Chapter 1

1. INTRODUCTION

Language modeling is technique of processing huge text data and designing a statistical based or syntactical based language model which can be used in many applications (Natural Language processing, spell checker, word predictor). It uses probability distribution and assign probabilities to each sequence of the word/sentence and, after that it selects the most relevant word sequence according to given input. Search engines makes great use of language modeling. It takes few letters of a word as an input and recommend list of most likely word/sentence which user is expected to type. This makes it easy for user to search for a topic or phrase. Google has prepared a language model by processing approximately 1 trillion words, which is used to provide recommendation for Google search engine [14].

In case of English text typing rate is nearly 35 wpm for keyboard [17]. If we look at the Hindi language the text typing rate is around 25 wpm with Standard keyboard [18], and 4-6 wpm with virtual keyboard. It is very less than typing rate in English, because in Hindi many character formation require more than one keystrokes to complete that character (like ऋ, क्ष etc.).

N-gram approximation is a technique used in language modeling, which computes the probability of n^{th} word by analyzing previous $n-1$ words using probability distribution. N-gram technique can be implemented according to requirement of the application (unigram, bigram, trigram, 4-gram… *etc*.) [4][20]. Unigram implementation stores each word independently with their word counts, hence probability of next word does not depend on any previous word.

Unigram is simplest implementation of N-gram technique and nor efficient neither useful in business applications but it require less memory and computation as compare to other N-gram implementation strategies like bigram in which the probability of n^{th} word is depend on the adjacent previous word means $(n\text{-}1)^{th}$ word it means the probability of same n^{th} word can be differ if $(n\text{-}1)^{th}$ word is different.

Bigram implementation is better than unigram and provides efficient result than unigram but it is not sufficient for most applications. To get more accurate result trigram implementation is used and efficiency of model can be improved further by using 4-gram and 5-gram

implementation, but trigram implementation provide reasonable accuracy and efficiency, so most of the application implements trigram based models.

Trigram implementation is complex than unigram and bigram. Here, probability of n^{th} word depends on $(n\text{-}2)^{th}$ and $(n\text{-}1)^{th}$ words. There are some techniques used to evaluate the performance and efficiency of language models. Perplexity is a method used to measure performance and efficiency of probability based language models. It states that a good language model should have less perplexity value, so we can conclude that the perplexity of trigram should be less than the bigram and unigram models.

1.1 Language Model for Word Recommendation System

There is mainly two strategies used to design language model for word recommendation system [11]. Statistical based and syntactical based language modeling.

1.1.1 Statistical language model:

Statistical language model produce accurate results and efficiency for given sequence of words, accuracy of result is based on training data and learning process of language model. Probability distribution is used to assign probability to next word, the probability of next word is completely depends on previous sequence of words, it means the probability of a word w_i get changed if we change the sequence of previous words. There is some issue with statistical based language model.

- *Sparsity*: In statistical model, many combinations of words does not appear as a sentence in practical, which means count for that word pair is zero, for large training dataset it leads to problem of sparsity in language model [1].
- *Unknown words*: Statistical model is trained on the training dataset which is normally set of millions of words, but in some cases it may happen that particular word does not belongs to training dataset. In that case language model is unable to calculate probability for that word sequence which has unknown word [6].

	किसान	कपास	को	प्रति	वर्ष	लगातार	उगाते	चले
किसान	0	28	20	18	8	16	4	2
कपास	11	0	6	1	1	6	8	0
को	14	8	0	6	5	3	2	0
प्रति	3	0	0	0	9	2	0	0
वर्ष	1	2	0	0	0	2	0	0
लगातार	0	2	0	0	0	0	1	0
उगाते	0	0	0	0	0	0	0	1
चले	1	0	0	0	0	0	0	0

Table 1. Sparsity Problem with bigram matrix

1.1.2 Structural language model:

Structural language modeling takes structure of word sequence also into account, which helps in producing more meaningful result than statistical based model [11]. It uses Part of Speech (POS) tagging, in which a tag is associated with every word of the sentence and categorize every word into a separate tag category. Tags in POS define a word class, which is cluster of similar type of words; every word belongs to a particular class and tag associated with that class does represent that word.

Unknown word problem of statistical model can be removed by using word classes and tagging, different word classes can be created by taking every possibility into account for that a word can be formed. For example, apart from all known word classes (like verbs, nouns, helping verbs, adjectives, pronoun, prepositions, abbreviation, numbers *etc*), we can create some classes forunknown words like (*capital_word, first_letter_capital, start_with_number, start_with_special_characteretc*) [27], this type of tagging makes language model more robust and it is also able to process unknown word sequences.

1.2 Language Modeling Terms

1.2.1 Linear interpolation:

Linear interpolation is a technique used to avoid sparsity problem in probability matrix. Linear interpolation compute mix probability by combining probability of $(n\text{-}1)^{th}$, $(n\text{-}2)^{th}$ up to unigram

with n^{th} gram probability, for example to calculate probability of a trigram sequence (w_1, w_2, w_3) it combine probability of its bigram (w_2, w_3) and unigram (w_3)and multiply each with a weighted multiplying factor λ_1[7].

$$q(w_1, w_2, w_3) = \lambda_1 * q(w_1/w_2, w_3) + \lambda_2 * q(w_2/w_3) + \lambda_3 * q(w_3),$$

Where λ_1, λ_2, λ_3 are weighted multiplying constants for trigram, bigram and unigram respectively and $\lambda_1 + \lambda_2 + \lambda_3 = 1$

The value of $\lambda_1$is calculated by taking a small part of training dataset called validation set anddefine $c(w_1, w_2, w_3)$ i.e. number of time the sequence w_1, w_2, w_3 occur in validation set and then select values of $\lambda_1, \lambda_2, \lambda_3$, such that it gives maximum value for the function,

$$L(\lambda_1, \lambda_2, \lambda_3) = \sum_{w_1, w_2, w_3} C(w_1, w_2, w_3)\ log\ q(w_1/w_2, w_3),$$

Where $q(w_1/w_2, w_3)$ is combination of maximum likelihood probability of trigram, bigram and unigram and can be define as,

$$q(w_1/w_2, w_3) = \lambda_1 * qML(w_1/w_2, w_3) + \lambda_2 * qML(w_2/w_3) + \lambda_3 * qML(w_3)$$

1.2.2 Smoothing:

Sparsity of N-gram count matrix in language model can be overcome by smoothing the occurrence of words. In smoothing, word count of N-gram sequence is reduced by a fraction and then total of the reduced fraction is known as “missing probability mass” [7]. This missing probability mass is assigned equally between all unseen sequence of words. By ensuring smoothing, language model never generates zero probability for unseen word sequences [30].

1.2.3 Discounting method:

It is a smoothing technique in which a constant fraction between 0 and 1 is subtracted from total counts of every word sequence of N-gram, which results in total probability sum of all N-gram sequences less than 1. The left over probability (missing probability mass) is then distributed over all the unseen word sequences equally [30].

Sequence(s)	Count(s)	Probability(s)
My, school	10	10/30
My, father	8	8/30
My, car	7	7/30
My, shoe	5	5/30
The, fan	0	0
The, box	0	0
Total	**30**	**1.0**

Table 2. Bigram count and probability with unseen sequence

After applying absolute discount of 0.5 from each bigram sequence (w_1, w_2), where count of bigram pair $C(w_1, w_2) > 0$. The resultant table will have some probability for the unseen pair instead of having zero probability.

Sequence(s)	Count*(s)	Probability(s)
My, school	9.5	9.5/30
My, father	7.5	7.5/30
My, car	6.5	6.5/30
My, shoe	4.5	4.5/30
The, fan	0	1/30
The, box	0	1/30
Total	**28**	**1**

Table 3 Count of unseen words after applying absolute discounting method

Table 2 showing actual counts for words and table 3 showing discounted counts which is calculated as $Count*(s) = Count(s) - 0.5$, here 0.5 is a constant weight subtracted from every N-gram sequence which has Count(s) > 0. After discounted weight the missing probability mass can be calculated as,

$$\alpha(w_{i-1}) = 1 - \sum_w \frac{Count*(w_{i-1},w)}{Count(w_{i-1})}$$

In our example $\alpha(w_{i-1})$ = 2/30, and the value of α is distributed equally between two unseen word sequence (*the, fan and the, box*) 1/30 each

1.3 Techniques Used in WRS

1.3.1 Chain rule of probability:

It is a probabilistic function based on markov chain. It is used to calculate the probability of particular letter or word sequence. On the other hand N-gram measure probability of next word based on list of existing words, In N-gram the list of existing words vary according to type of N-gram used (Unigram, Bigram, Trigram *etc*) [26]. A word recommender system with efficient language model can play a major role in area of text processing like personal letter writing, report writing and many other text processing tasks.

It is a probability based approach, it estimate the chances of next variable by multiplying the probabilities of previous variables till the present variable.

Consider an indexed set $A_1, \ldots\ldots, A_n$ to find the value of member of joint distribution,

$$P(A_n, \ldots\ldots, A_1) = P(A_n|A_{n-1}, \ldots\ldots, A_1) * P(A_{n-1}, \ldots\ldots, A_1)$$

With four variables, the probability of $P(A_4)$ can be calculated as,

$$P(A_4, A_3, A_2, A_1) = P(A_4|A_3, A_2, A_1) * P(A_3 \mid A_2, A_1) * P(A_2|A_1) * P(A_1)$$

1.3.2 N-gram approximation:

N-gram are a consecutive series of fixed length "*n*" characters which are subset of a word or sentence, a N-gram can also viewed as part of two words. Unlike stemming which tries to find the stem of word that should represent a semantic meaning, N-grams does not take this theory in account, N-gram does not represent semantic meaning of the word. Some N-gram allow inter-word symbol to be part of N-gram with *N* greater than two[25].

```
For example,
Sentence- What is your name
Bigram  - wh, ha, at, is, yo, ou, ur, na, am me
Trigram - wha, hat, is#, you, our, nam, ame(without inter-word symbols)
Trigram - #wh, wha, hat, at#, #is, is#, #yo, you, our,ur#, #na, nam, ame,
me#(with inter-word symbols)
Where # represents inter-word symbol which is belongs to any of the (period,
semicolon, blank, colon etc.)
```

Trigram analysis provides a feasible data structure for identifying transposed and misspells characters, it can correct potential input errors also by insertion, substitution or deletion of character. Frequency of N-gram pattern occurrence can also used to indentify language of an item.

Error Category	Example
Single Character (Insertion)	Coputer
Single Character (Deletion)	Compputer
Single Character (Substitution)	Compater

Table 4. Error correction Example using N-gram

N-gram approximation rule for trigram,

$$Count(w_i/w_{1-2,}w_{i-1}) = \frac{Count(w_{i-2,}w_{i-1},w_i)}{Count(w_{i-2,}w_{i-1})}$$

For example, we have three sentences here,

<s> I am prem</s>

<s>prem I am</s>

<s>I do not like green tea</s>

The Bigram approximation rule can be defined as,

$$Count(w_i/w_{i-1}) = \frac{Count(w_{i-1},\ w_i)}{Count(w_{i-1})}$$

Now, probability for the word sequence can be calculated as,

P(I/<s>) = 2/3 = 0.67 P(prem/<s>) = 1/3 = 0.33

P(</s>/prem) = 1/2 = 0.5 P(prem/am) = 1/2 = 0.5

P(am/i) = 2/3 = 0.67 P(do/i) = 1/3 = 0.33

1.3.3 Levenshtein distance*:*

It is an edit distance method used to find the distance between two given string, it is an efficient distance calculating method and helpful in spell checking applications. It takes two strings and compares each character of string with characters of other string. Levenshtein distance performs basically three operations to check the gap between similarity of two strings (insertion, deletion and substitution) [9].

First string	Second string	Operation	Edit distance
Setting	Hitting	2 substitution	2
Check	Lacks	1 deletion 1 insertion 2 substitution	4

Table 5. Levenshtein distance for two strings

Weight for operations (insertion, deletion, substitution) is generally 1 but based on requirement we can choose different weights for these operations. In word recommendation system, user can make spelling errors and sending the wrong spell word direct to language model for processing will result in different output than expectations so before processing input word, a preprocessing system should apply Levenshtein distance on user input and stored data to remove the spelling mistakes from input words.

1.3.4 Basic Introduction of WRS:

The project is designed in two layers:

- Application Layer
- Presentation Layer

Layered approach is used keep related task separate from each other and modularization within layers so that modules can work independently, it increase reusability of programs. A well defined function can work independently and can easily modifiable without affecting other modules or layers of the project.

We have design this project in Netbeans IDE using java technology, we have used swing for graphical user interface. It is complex for unskilled operators and regular user to use standard keyboard for Hindi typing purpose, we have designed a virtual keyboard for those users.

Execution of project starts with loading all datasets into main memory. After that presentation layer make first move by interacting with user through keyboard and mouse, presentation layer sends input data to application layer where actual processing of word recommendation performed.

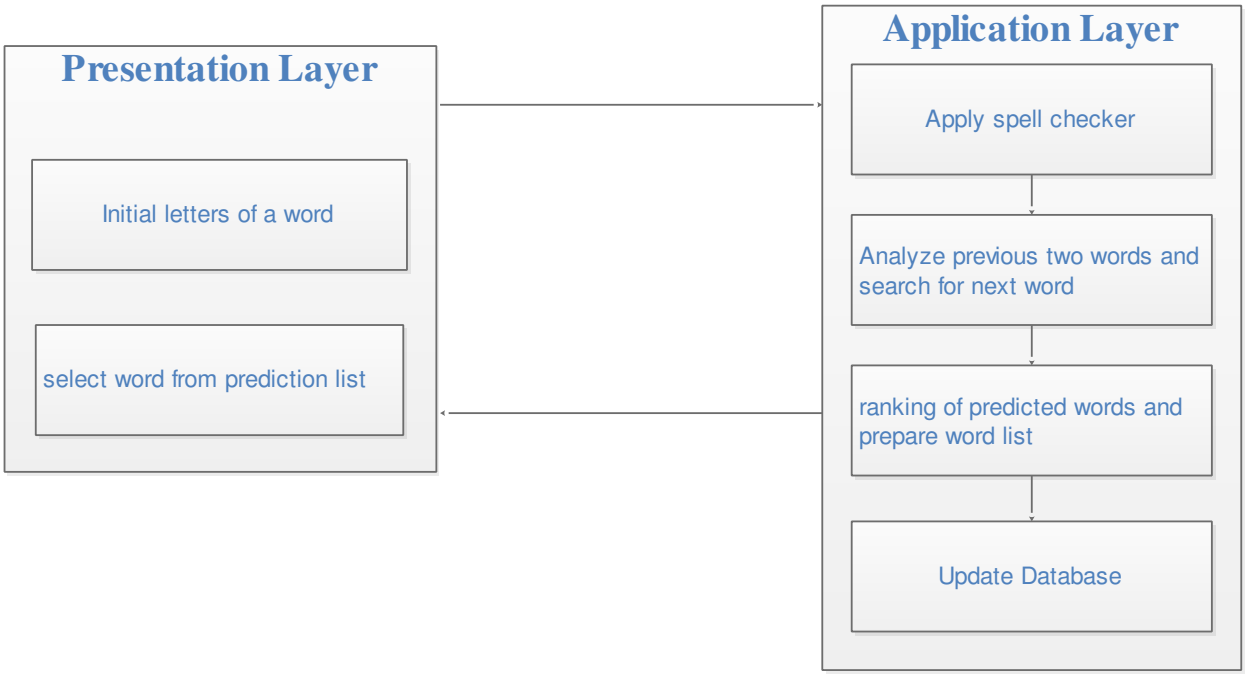

Figure 1.Showing Basic model of Proposed Model

Presentation layer perform mainly two task, one is take user input then parse it and then pass it to application layer and other is to take result sent by application layer and parse it again to show on output screen to the user. The application layer contains utility classes and other classes to perform computations logic.

Chapter 2

2 LITERATURE SURVEY

An efficient word recommendation system require a large set of training data (at least a million words) and to process and store these large scale data selection of right data structure is a crucial task, because the performance of system depends on the type of data structure used. Denial and Jan discussed comparisons between some mostly used data structures to process 2-grams, 3-grams up to 5-grams [24]. They have made comparisons on hash table, B+ tree, ternary AVL tree, doubly ternary AVL tree.

Hash table is a <*key, value*> pair based data structure in which value is stored with respect to particular key, key in hash table should be unique so that every value stored in data structure can be accessed uniquely. The main advantage of hash table is that it takes constant time to find any value stored in hash table using its key [13]. In hash table each value has a unique key and the key is nothing but a value generated by a hash function during insertion in hash table, the hash function apply a particular computation to obtain a value which is a key and this value should be identical, it means for every insertion in hash table the hash function should generate a unique value. So the main task in hash table is to select a good hash function, because duplicate key for different values cause collision during insertion. Collision cause when hash function generate same key for two different values and it require extra effort to resolve collision in hash table [24].

There is mainly two techniques to deal with collision, open addressing and chaining. Open addressing provide a alternative position for the value to be stored it require the data structure to have some extra space to store collide values. Open addressing has several strategies to select alternative position for values.

Linear probing quadratic probing and double hashing are strategies used in open addressing, linear probing provide very next available position from collide position which result in concentration of data and frequent collision, quadratic probing a quadratic polynomial value is added with hash key to store data, it avoid concentration of data but still not efficient [13]. Double hashing is a good alternative of linear hashing and quadratic hashing, a well defined hash

function is applied to compute the alternative position for data. Double hashing is efficient collision resolve method but it require another hash function computation to store data. So selection of a good hash function can make hash table more efficient and fast by reducing occurrence of collision.

B+ tree data structure store data in tree structure where every node has some keys and pointer to the child nodes of the tree, data in B+ tree stored at leaf node and each leaf node is chained to each other by linked list. The size of a B+ tree node can be chosen according to block size of hard drive so that at run time system can fetch a complete block from hard drive to main memory.

Binary search tree (BST) is a tree data structure. It consists a node with data and two pointers to hold the left sub tree and right sub tree, the property of BST can be defined as, the elements in left sub tree of BST should be less than or equals to the node and elements in right sub tree should have elements with higher value than the node of sub tree. This property of BST reduces the searching time complexity to $O(\log n)$ by looking up only in left or right sub tree of the node.

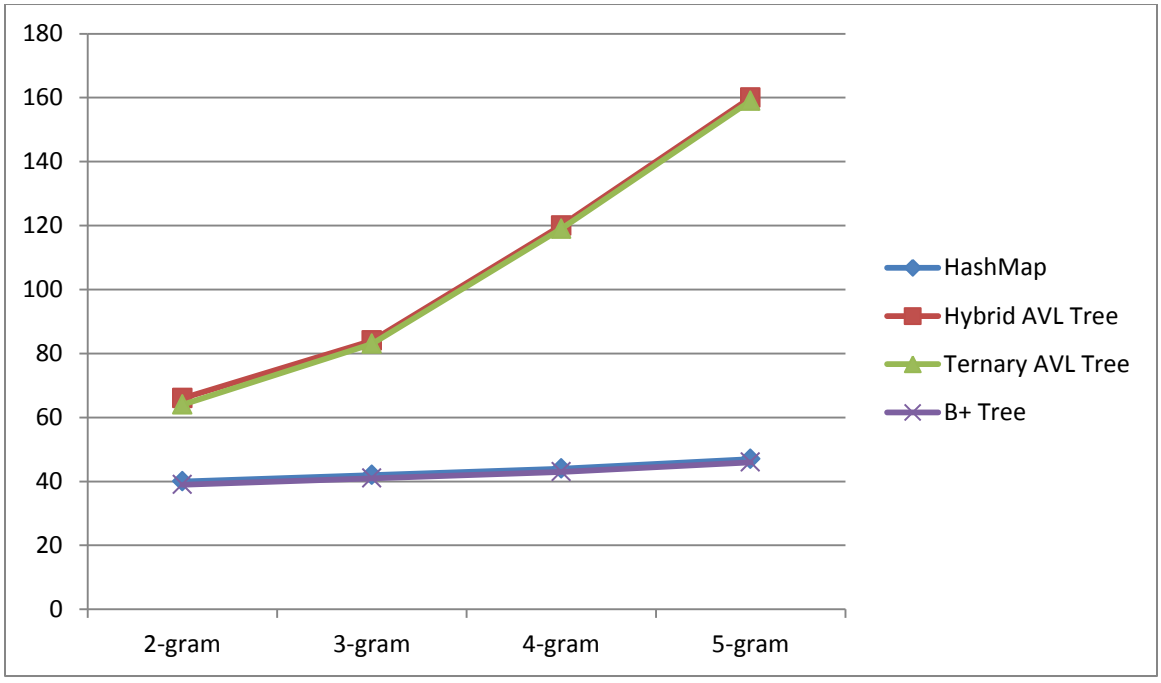

Figure 2. Comparison of memory consumption [24]

In some situations BST perform like linked list and create chain during tree creation instead of tree. For example if elements needs to store in BST are sorted in ascending or descending order then BST will form a chain and behave like a linked list which result in $O(n)$ searching complexity.

Unbalanced tree formation has worse time complexity than balanced BST. This problem can be solved using AVL tree, AVL tree is a BST which perform a rotation when tree goes unbalanced. The last thing need to add with AVL tree is a ternary search tree, ternary search tree is a tree which has same property as binary tree but instead of two pointers for left and right sub tree in BST ternary search tree has three pointers two for left and right sub tree and third pointer is to point the next ternary tree which store the next element. In big scenario ternary search tree can also be unbalanced so the combination of TST and AVL tree can be used as a Hybrid data structure for N-gram it keeps the searching complexity for data to $O(logn)$ [24].

Trie is a data structure mainly used to process strings, it is a tree type data structure in which each node contains three information, first is data value, its occurrence and an array of pointers which points to the node of all data elements which comes after that data value. Trie is mainly used with strings, time complexity for trie is better than BST searching in trie take $O(m)$ time where m is the length of input string. It is efficient in term of speed but a large part of memory is wasted in storing pointer of array with every node of trie.

For example if we have a trie data structure to store alphabetical string in it, then we need an array of pointer of size 26 one for each alphabet but in practical many combination of alphabets never occur and only a part of array will have values within it but we have created memory for all alphabets so remaining array position will remain empty. This drawback of trie makes it hard decision to select it as a data structure for such a huge data set of strings.

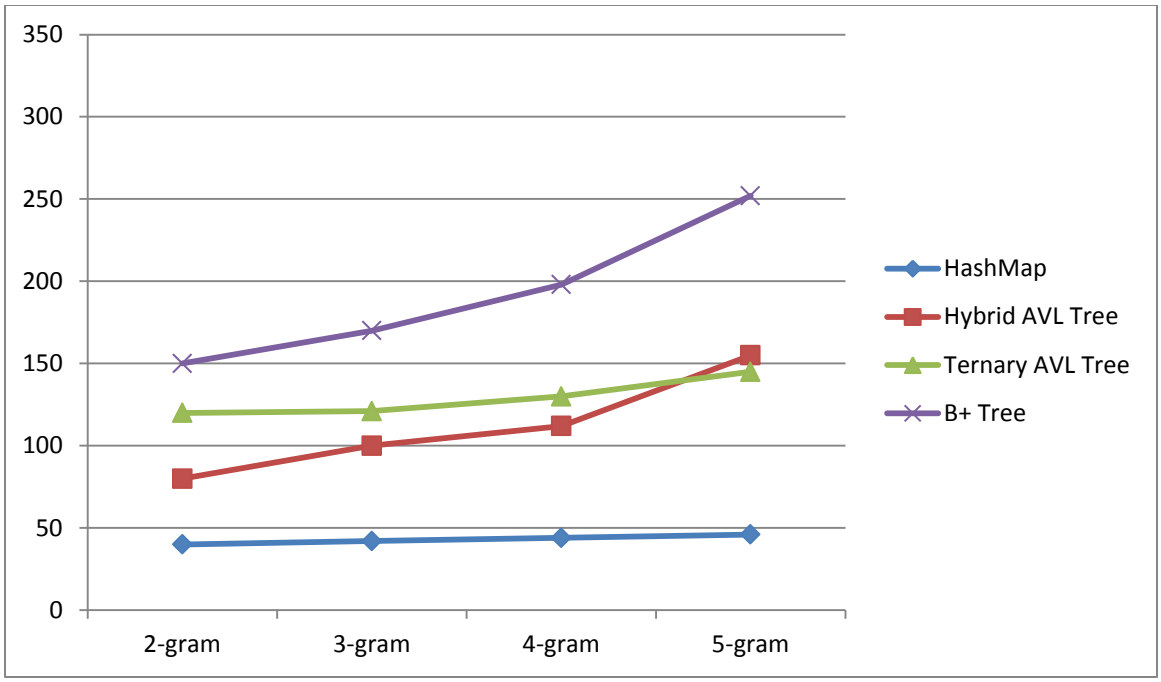

Figure 3. Comparison of search time [24]

Conclusion of the study shows that hash table is efficient then other data structure in context of time complexity to search elements. In case of space complexity Doubly ternary AVL tree is efficient then hash table by a significant difference but hash table can also be considerable [24].

Google WebIT has 4 billion N-grams for 13.5 million words. The language model to manage 4 billion N-grams is a challenge and at the same time decoding a single sentence. So speed is also critical. 4 billion N-grams are stored with associated counts in 10GB memory which is smaller than "state-of-art" lossy language model implementation [14].

In WebIT corpus, the most frequent N-gram occurs about 95 billion times[14]. Storing this count explicitly it require 37 bits to store an encoded an N-gram, but as noted by Guthrie and Hepple, there is only 77000 unique counts of corpus [15]. So we can encode all counts using 20 bits and can stores all unique corpus in an array [23]. It saves 17 bits per N-gram which reduced memory storage from 16GB to 9GB for WebIT.

On the other hand lack of computing devices in Hindi and complexity of writing words in Hindi are some major factor to a digital divide for writing in Hindi, because formation of an akshara in Hindi can be formed by using combination of consonant *'C'* and vowel *'V'* as *(C+V,*

C+V+V), C+C+V) [18]. 19's literatures in word prediction system have used only frequency of words to estimate the most desire word which is based on the unigram frequency of the word without considering the recent text entered by the user into account [27].

With the improvement in statistical models most of word recommendation systems calculate bigram and trigram probabilities to make prediction more useful and some additional parameters can be added as score matrices to make prediction more precise [25]. A common problem with system with unigram, bigram and trigram is that system does not have recommendation for unknown word sequence and it gives zero probability in such scenario [25].

Some statistical models and methods can be used to eliminate null count problem with bigram and trigram models, linear interpolation, discounting methods and HMM of n-gram can reduce this problem [10]. Part-of-speech (POS) tagging is an art of associate a well-defined label with every word in the dictionary and these labels are processes to find desired words [10].

Some researcher combined Part-of-Speech (POS) with statistical N-gram model for Hindi to get more accurate results. Though in Hindi, it is not complete solution to associate a word with only one category of tag because some words in Hindi belongs to more than one word class of tagging but for large corpus dataset we can ignore this case. Part-of-Speech tagging tags every word of dataset with a particular tag and implements N-gram for tags instead of words. The sequence of POS tagging trigram then used to predict the next tag which can occur after that tag sequence, the use of tag based N-gram work efficient with unseen word occurrence in input data, because language model can categorize unknown words into a separate word class category.

2.1 Some Features of POS Tagging

Context-based features: for any word the context consists of POS tag of previous word that is unigram, Combination of POS tags of last two words i.e. bigram, current word and the next word predicted by the language model by processing the given sequence of word [10].

Word features capture property of word being tagged [10]. It includes,

- Suffixes: if word suffix is similar to a given suffix.
- Digits: the word is complete numeric or has any digits with it.
- Special characters: Does word contain any special character like '_' within it.
- English word: Occasionally occurrence of English word in Hindi text should be handled.

Corpus based features: this feature rely o information extraction from corpus data. Which are;

- Corpus word belongs to a single tag
- All possible tags of current word, as seen in corpus data.
- Does word occurred with only singe tag in corpus data.
- All possible tags of next word, as seen in corpus data.

Research in area of language modeling for word prediction system has been done by many individuals and firms and they have made good use of the language model for word recommendation system. In case of Hindi language, researcher has put their effort to make a good language modeling by using different techniques but still there is space for improvement in Hindi language model. A main reason behind that is the unavailability of sufficient resources and less interest of organizations in Hindi as compare to English. There is no standard vocabulary for Hindi which give all needful information about Hindi language words, verbs, punctuations, preposition *etc.*

M.K. Sharma and DebasisSamanta worked on Devanagari Script, As compared to English, Hindi language has larger character set including (13 vowels, 12 matras, 33 consonants, some special symbols like *anusvara*, *chandrabindu*, *visarga*, and *nukta* etc.) [19][22], apart from that there are some phonetically and graphically similar characters which increase the chances of confusion during selection of character from database using language model and sometime leads to recommend wrong word [25].

Their result shows that using POS tagging with N-gram approximation the system deliver good performance with an average accuracy of 88.40% with the accuracy of 89.35% [10]. Some of the POS tags detected most precisely like PRP (personal pronoun with 98.19%), SYM (symbol with 97.58 %), CC (coordinating conjunction with 91.59%).

Number of words tagged	9154
Number of words wrongly tagged	975
Correctness Accuracy (%)	89.3489

Table 6. Accuracy of POS tag classes [10].

Average word length in Hindi is 4.69 as compared to 5.1 in English [3]. Huge entries in lexicon with lower average word length play a role in performance of word recommendation system [29]. These issues make text composition task more error prone and time inefficient and finally result in poor text entry rate. Apart from choosing technique for language model Hindi language itself has some issues which needs to take into account while designing a model for it.

2.2 Issues in Language Model of Hindi

2.2.1 Large character set:

As discussed in above paragraph, Hindi language has large character set and characters can be classified in different categories but it is not the problem, the standard keyboard is not flexible while typing in Hindi, formation of a words may need keys other than alphabetic, numeric and *shift* key. In Hindi typing *Ctrl*, *Alt* key are also used with alphabetic, numeric and *Shift* key which reduces the overall typing rate by increasing total number of keys needs to type a word [21].

2.2.2 Input sequence:

Devanagari script has a way in which user has to type keystrokes to form desire word. For example the type the word निर्मित, user has to type it into this order; न+ि + र+् + म + ि+ त (nirmit). For general user it is hard to remember all this sequence in which these keys should be typed to form the word [16][21][25].

2.2.3 Normalization:

Dataset is required to be normalized in case of Indian language Hindi [5][8]. There exist some characters in Hindi having equivalent Unicode representations. For example the Unicode sequence of alphabet क्ष (U0915 + U094D + U0937) can also be represented in different forms such as क् +ष and its unicode sequence is (U0915 + U094D + U200C + U0937) and क्षhas Unicode sequence (U0915 + U094D + U200C + U0937) [constable 2004]. According to that if normalization is note done properly then searching for a word written in one form can omit the words in another form [5]. It will reduce the hit rate in word recommendation system.

Normalized form		Combining sequence	
Character	Unicode	Combination	Equivalent Unicode
ख़	U0959	ख + NUKTA	U0916 + U093C
ड़	U095C	ड + NUKTA	U0921 + U093C
फ़	U095E	फ + NUKTA	U092B + U093C
ज़	U095B	ज + NUKTA	U091C + U093C
क्ष	U0915 + U094D + U0937	क+ ् + ष	U0915 + U094D + U0937
क्ष	U0915 + U094D + U0937	क+ ् + ZWJ + ष	U0915 + U094D + U200D + +U0937
क्ष	U0915 + U094D + U0937	क+् + ZWNJ + ष	U0915 + U094D + U200D + U0937

Table 7. Words with similar Unicode values having different representations [25]

2.2.4 Typographical variants:

In Hindi many words can be represented with more than one spelling, like हिंदी can also be represented as हिन्दी, these both different forms of same meanings are correct and acceptable in Hindi language. It is important task of language model to find similarity between these types of words and process them as a same word [2]. Some more example can explain the necessity of understanding the similarity between these type of words म्यांमार can also be represented as म्यान्मार and similar to these लिए and लिये has same meaning and purpose

2.2.5 Phonetic similarity:

Some alphabets in Hindi are phonetically similar to each other and can make confusion in user mind to select the correct one like स, ष andश sound almost similar but it is up to language model to select the right alphabet [12].

Chapter 3

3 PROBLEM STATEMENT

Growth of internet in last decades indicate that world population is going towards digitalization. Every sector is adopting digital documentation. This includes report writing, letter writing, project documentation, health records and e-papers. Usability of digital documentation requires an efficient way of writing documents and information retrieval. A word recommendation system (WRS) provides a platform of writing fast and efficient documents with minimum error.

In case of English text typing rate is nearly 35 wpm for keyboard. If we look at the Hindi language the text typing rate with keyboard is around 25 wpm with Standard, and 4-6 wpm with virtual keyboard. It is very less than typing rate in English, because in Hindi many characters require to type more than one keystrokes to complete a character (like ऋ, क्ष etc).

A WRS for Hindi should be implemented with following functionalities:

1. WRS for Hindi should provide recommendations to user for particular word sequence which can increase the typing rate.
2. WRS should also provide a virtual keyboard for typing in Hindi, which should be efficient in terms of usage of frequent letters in the middle of the keyboard.
3. WRS should enable user to type just initial letters of a word and predict the complete word, so that user can get required word without typing all keystrokes needed to type it.
4. It should also be able to save some keystrokes for each word by recommending relevant and required word to user and can also reduce chances of error in report, letter writing.
5. Since there are no online spell checkers for Hindi, the system should be able to suggest correct word even if the user has misspelled.

Chapter 4

4 IMPLEMENTATION

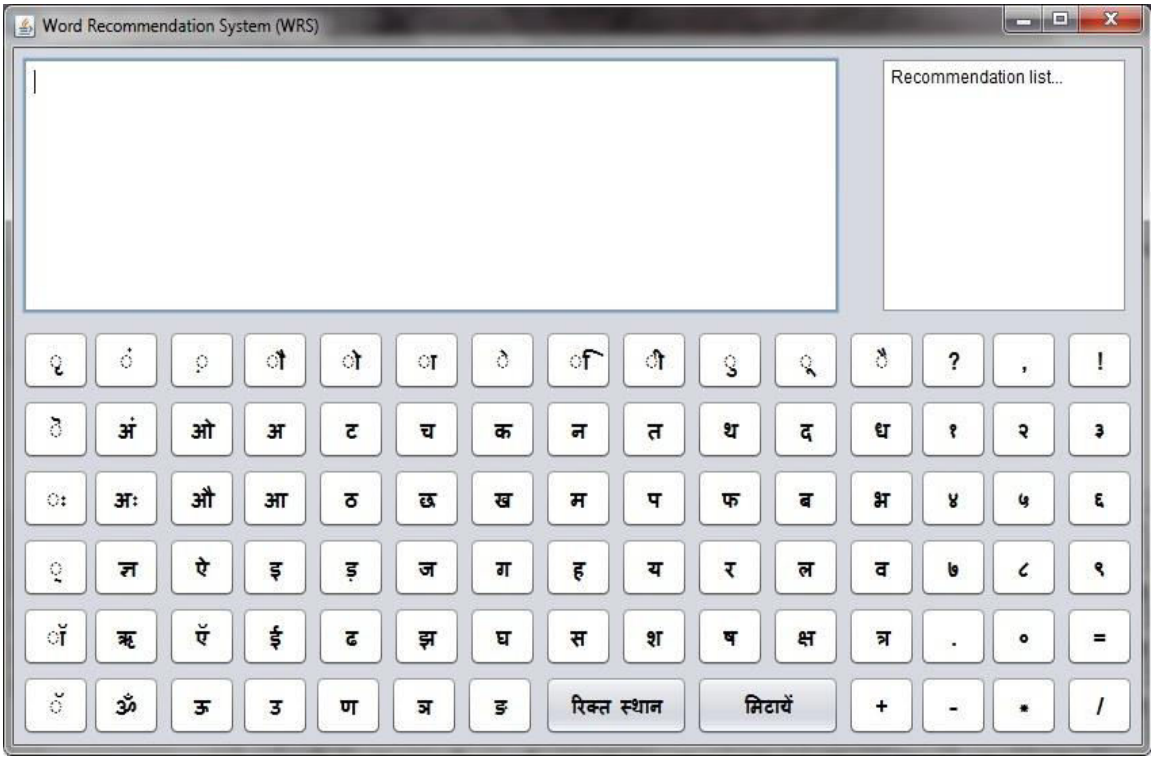

Figure 4. User Interface for WRS

Above figure is graphical user interface (GUI) for the project. We have used Swing components and designed a virtual keyboard using buttons, each button represents a single alphabet of Hindi language, the virtual keyboard provide a simple interface for user to type in Hindi. It is observed that text typing rate using virtual keyboard is less as compare to hardware keyboard [28],but we have tried to maximize it.

We have analyzed corpus data and tried to find out the best way for positioning of alphabets on GUI so that the user makes minimum movement of mouse pointer to form a word. Main drawback of virtual keyboard is that, it requires frequent movement of mouse because if alphabets require to complete the word is at distance from each other than a large part of time is wasted in mouse movement. We have found that some words न, ह, म, सoccur frequently in a word hence we have placed these words in the middle of keyboard so that user need to make

minimum mouse pointer movement to reach to these alphabets. Same study is done with *vowels, matras* and *numbers*, we have tried to place them at right place so that word can be formed efficiently. Some characters in Hindi used rarely like ॐ, ऋ, ज्ञ, त्र, क्षhence we have place them on corners so that more useful alphabets can be placed on center.

The system is built to predict the next word by analyzing few previous words. N-gram approximation is a technique which is efficient for language modeling, some major IT organizations like (Google, Microsoft) uses N-gram technique to design a language model for word prediction and speech recognition. To predict next word*(n)* for particular sequence of word N-gram analyze previous*(n-1)* words and predict the next best match word for the sequence to user, how many previous word should be analyzed is depends on the level of N-gram implemented.

N-gram can be implemented at simple level to complex level according to the need of project. A simple N-gram can be implemented as unigram and bigram, in unigram only current word is stored and to find best matching word for sequence does not require any analysis of previous words. In bigram best suitable next word*(n)* for sequence is selected by language model by analyzing only last word of the sequence, bigram implementation perform better than unigram because it involves last word to predict next word but still bigram cannot provide desired accuracy for normal language processing application.

Trigram implementation can be considered as baseline for good language model, it require to process previous two words to predict next word which provide considerable accuracy and it is efficient and less complex as well. For more accurate results N-gram implementation level can be increased to 4-gram and 5-gram, these provide more accurate results than trigram but have complex implementation structure over bigrams and trigrams. At the same time the memory requirement of 4-gram and 5-grams is more than trigram and require more computation and processing. So for medium level application, trigram implementation is a good choice because it is memory efficient and requires time efficient as well.

N-gram approximation is a technique used in Language Modeling which computes the probability of n^{th} word by analyzing previous *n-1* words.

For example,

$$Count(w_n/w_1 w_2, \ldots, w_{n-2} w_{n-1,}) = \frac{Count(w_1, w_2, \ldots\ldots, w_{n-1}, w_n)}{Count(w_1, w_2, \ldots w_{n-2}, w_{n-1})}$$

This section will discuss the proposed model for word recommendation system of predicting words when user press some keystrokes using virtual keyboard or actual keyboard. The section is divided into three subparts, 1. Design a language model 2. Predict word based on given input by user and recommend a list of words to the user.

4.1 Design of Language Model

The Hindi language, which follows Devanagari script, contains a large set of characters (13 vowels, 33 consonants, 12 matras and special symbols like *anusvara*, *chandrabindu*, *etc*.). To build a language model we have consider a Hindi corpus of IIT Bombay, our aim is to implement unigram, bigram and trigram from the words in input file. Data in file is in the form of sentences. We processed a sentence at a time to build unigram, bigram and trigram for that sentence.
Bigram formula can be expressed as,

$$Count(w_i/w_{i-1}) = \frac{Count(w_{i-1},\ w_i)}{Count(w_{i-1})} \quad (1)$$

Trigram formula can be expressed as,

$$Count(w_i/w_{1-2,} w_{i-1}) = \frac{Count(w_{i-2,} w_{i-1}, w_i)}{Count(w_{i-2,} w_{i-1})} \quad (2)$$

One practical issue with bigram and trigram model is, many word sequence form bigram and trigram never occur during training set. It means we have zero count for many columns of particular word. It increases the size of language model which result in reduced efficiency for language model, so if a bigram or trigram word has zero count for particular word then why to store that entry. Hence to address the problem of sparseness in model we have used hashmap data structure to build unigram, bigram and trigram, using hashmap we can completely remove the problem of sparsity in case of bigram and trigram.

The use of hashmap introduce a new problem of No Prediction for Unknown words, it means if a word sequence is unknown for language model then system can not recommend word for that sequence, so to address that problem we have build bigram and trigram for Part-of-

Speech (POS) tags, in case if word sequence is unknown to system then it switch to simple trigram/bigram to POS trigram/bigram.

Bigram for Part-of-Speech (POS) tags is:

$$Count(POSw_i/POSw_{i-1}) = \frac{Count(POSw_{i-1},POSw_i)}{Count(POSw_{i-1})} \quad (3)$$

Trigram for Part of Speech tags,

$$Count\left(POSw_i/POSw_{1-2,}POSw_{i-1}\right) = \frac{Count\left(POSw_{i-2,}POSw_{i-1},POSw_i\right)}{Count\left(POSw_{i-2,}POSw_{i-1}\right)} \quad (4)$$

Suppose In Hindi (आज सोमवार है) ==➔ POS(आज/PNN, सोमवार/NN, है/HV) In that case particular tags are assigned to each word, like आज is considered as Pronoun, सोमवार is considered as a Noun and है is considered as a Helping Verb. Now these tags shows word composition and these tags are processed to implement POS tags trigram.

4.2 Recommendation Methodology

As user type initials of a word (w_i), system will pass last two words (w_{i-2}, w_{i-1}) with w_i in a sequence of w_{i-2}, w_{i-1}, w_i to prediction module because our language model is design to process words at trigram level and need two previous words to predict most appropriate word(w_i). When user start typing the very first word previous words w_{i-2} and w_{i-1} will be empty, in that case prediction module accepts only w_i as a processing input and predict word list accordingly.

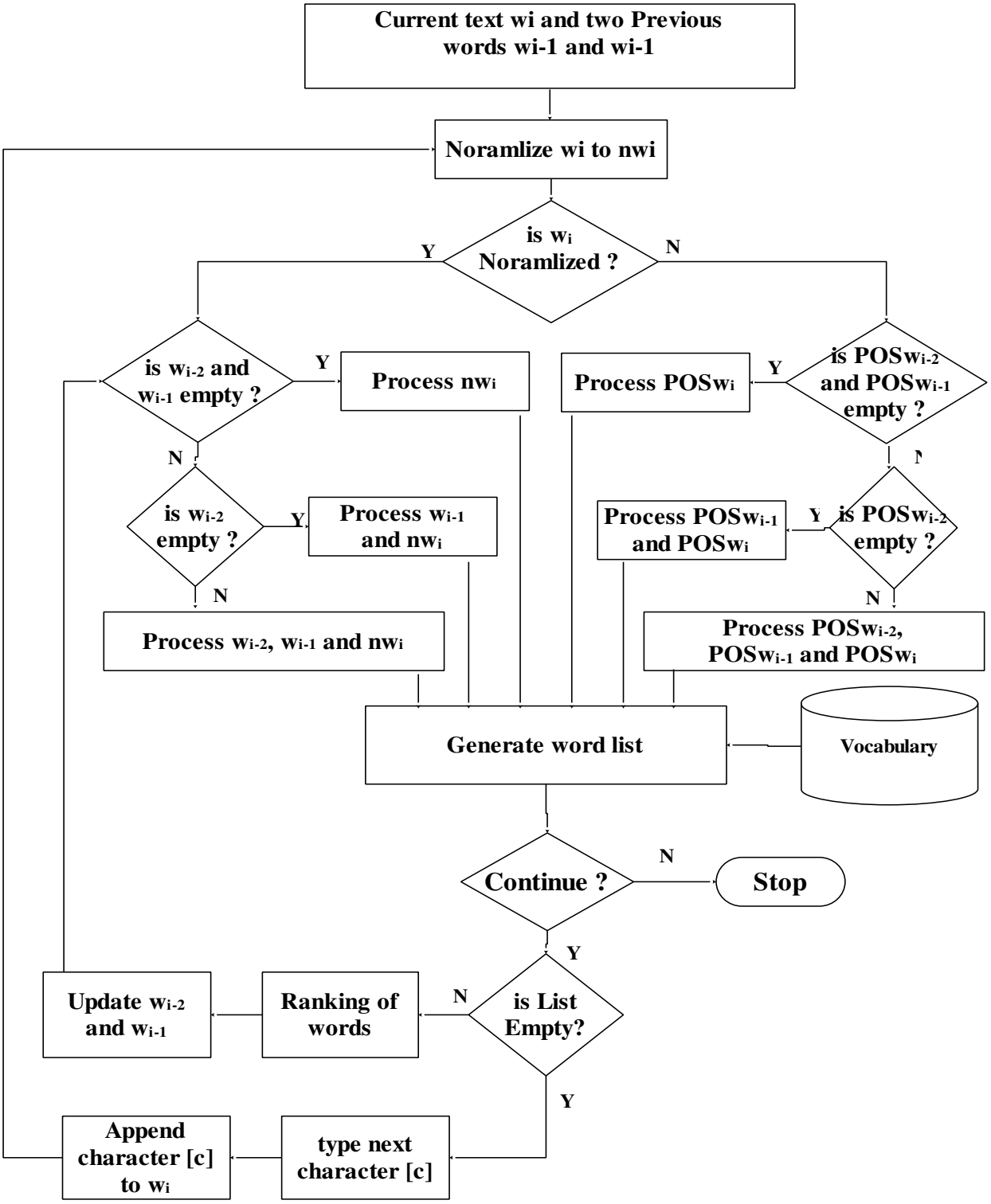

Figure 5. Flow chart of proposed model

4.2.1 Normalization of input:

In Hindi, we have some word having different representation but point to same meaning. These confusing words can affect accuracy of result, like (हिंदी, हिन्दी) and (लिये, लिए) produce same meaning. There exist some characters in Hindi having equivalent Unicode representations. For example the Unicode sequence of alphabet क्ष (U0915 + U094D + U0937) can also be represented in different forms such as क् +ष and its unicode sequence is (U0915 + U094D + U200C + U0937) and क्षhas Unicode sequence (U0915 + U094D + U200C + U0937) [25].

Other reason for normalization is spelling errors. If input w_i does not match with any prefix of words stored in vocabulary then w_i has to normalized into nw_i using Levenshtein Edit Distance algorithm, a threshold value is defined for Edit Distance algorithm to avoid unknown nouns, if Edit Distance is more than threshold value then word $w_i$will not normalized to nw_i. In that case the sequence of word is passed to another module POS module which is described in POS module.

If user input w_i does not match with any prefix of words stored in vocabulary then w_i has to normalized to nw_i using Levenshtein Edit Distance algorithm and a threshold value is defined for Edit Distance algorithm, if Edit Distance is more than threshold value then word w_i will not normalized to nw_i. In that case the sequence of word is passed to another module POS module which is described in POS module.

ALGORITHM 1. FOR NORMALIZATION OF INPUT

normalize_input(input : String)

```
words_array[] : String
distance : int
previous_distance : int
norm_input : String
for i = 0 to lengthOf(words_array)
    distance = edit_Distance(input, words_array[i]) // compare input with every word
    if(distance<previous_distance)
norm_input = word_array[i]
previous_distance = distance
    :end if
:end for
return norm_input;
:end
```

edit_Distance(input1 : String, input2 : String)

```
length1=lengthOf(input1)
length2=lengthOf(input 2)
distance_array[length1][length2]
for I = 0 to length1
        distance_array[i][0] = i
:end for
for j = 0 to length2
        distance_array[0][j]=j
:end for
for i = 1 to length1
        for j = 1 to length2
distance_array[i][j] = minimum(distance_array[i-1][j] + 1, distance_array[i][j-1],
            distance_array[i-1][j-1] + (input1.charAt(i-1) == input2.characterAt(j-1)) ? 0 : 1))
        :end for
:end for
return distance[length1][length2]
:end
```

4.2.2 Word processing module:

After normalization of word w_i into nw_i system will check for previous two words and consider w_{i-2}, $w_{i-1}$and nw_i as a trigram to generate word list, if $w_{i-2}$and w_{i-1} are not present then system prepare word list by processing the normalized word nw_i. On the other hand if system found only w_{i-2}then w_{i-1} with $nw_i$is processed to generate word list. After normalization of word w_i into nw_i system will check for previous two words and considerw_{i-2}, w_{i-1} and nw_i as a trigram to generate word list, if w_{i-2} and w_{i-1} are not present then system prepare word list by processing the normalized word nw_i. On the other hand if system found only w_{i-1} then w_{i-1} with nw_i is processed to generate word list.

ALGORITHM 2. FOR WORD TRIGRAM PROCESSING

process_wordTrigram(key1,key2,key3)

```
key1 : first word
key2 : second word
key3 : third partial word // key3 may not available some times
trigrams : HashMap
bigrams : HashMap
unigrams : List
sum : int  // total sum of all words
p_score : float  // to store probability of words
bigrams = trigrams.valueOf(key1)
unigrams = bigrams.valueOf(key2)
for i = 0 to sizeOf(unigrams)
    sum = sum + unigrams.valueAt(i).count
:end for
if(isEmpty(key3))
    unigrams = unigrams.comparePrefix(key3) // all words having key3 as a prefix
:end if
for i = 0 to lengthOf(unigrams)
prob_array[i] = computeProbability(unigrams.valueAt(i), unigrams.valueAt(i).count, sum)
:end for
return prepareList(unigrams, prob_array)
```

compute_Probability(word, count, total)

```
probability : float
probability = (count / total) * (word.tag.count/ tag_size)
return probability
:end
```

Algorithm 2, describe concept behind generation of word prediction list, hashmap is used to store unigram, bigram and trigrams for the corpus dataset. During trigram construction,

sequence of three word chained to each other in similar sequence using hashmap. Bigram sequence is extracted from trigram hashmap using first word (key1) then same process is apply on bigram to extract unigram using second word (key2) after that p_score is calculated for each word. To increase accuracy of result POS trigram probability for given sequence is combined with word trigram probability.

4.2.3 Part of Speech tags Module:

POS module uses POS tag trigram to generate recommendation list. The functionality of POS tag based trigram is identical to word trigram. The only difference is in POS tag based trigram is that tags associated with words are used as input elements to predict next word instead of words. For example, next probably word for input (आजसोमवार) can be calculated as:

In POS module process tags associated with each word and compute probability for most likely tag sequence. Probability can be computed as,

$$Count\left(POSw_i / POSw_{1-2,} POSw_{i-1}\right) = \frac{Count\left(POSw_{i-2,} POSw_{i-1}, POSw_i\right)}{Count\left(POSw_{i-2,} POSw_{i-1}\right)} * \frac{POSw_i}{\sum POSw_i} \quad (5)$$

आज सोमवार ==➔ (POS)(आज/Pronoun,सोमवार/Noun), in that case system search for tags which can comes after the sequence (Pronoun and Noun) and according to the resultant tag list, system generate a list of words associated with those tag. POS module uses POS tag trigram to generate recommendation list. The functionality of POS tag based trigram is identical to word trigram module. The only difference is that instead of words, tags associated with words are used as input to predict next word. Different word classes are created and a unique tag is assigned to each class. Every word of dataset is categorized in one of the word class. Though some words in Hindi produce different meaning in different course like अच्छा (accha), it can be used as an adjective but it can also be used as an exclamation but these kinds of issues can be ignored for large dataset having millions of words.

ALGORITHM FOR Part-of-Speech TRIGRAM PROCESSING

process_POS_Trigram(key1,key2)

```
key1 : first word
key2 : second word
tag1 = getTag(key1)
tag2 = getTag(key2)
POS_trigrams : HashMap
POS_bigrams : HashMap
POS_unigrams : List
sum : int  // total sum of all words
prob_array : float  // to store probability of words
POS_bigrams = POS_trigrams.valueOf(key1)
POS_unigrams = POS_bigrams.valueOf(key2)
for i = 0 to sizeOf(POS_unigrams)
     sum = sum + POS_unigrams.valueAt(i).count
:end for
for i = 0 to lengthOf(POS_unigrams)
prob_array[i] = (tag.count/ total_tag_size)
:end for
return prepareList(unigrams, prob_array)
:end
```

5 RESULTS

We have implemented this project on a 64 bit windows 7 computer system having 2 GB of RAM and i3 second generation 1.80 GHz processor. We have design this project in Netbeans IDE using java technology, we have used swing for graphical user interface.

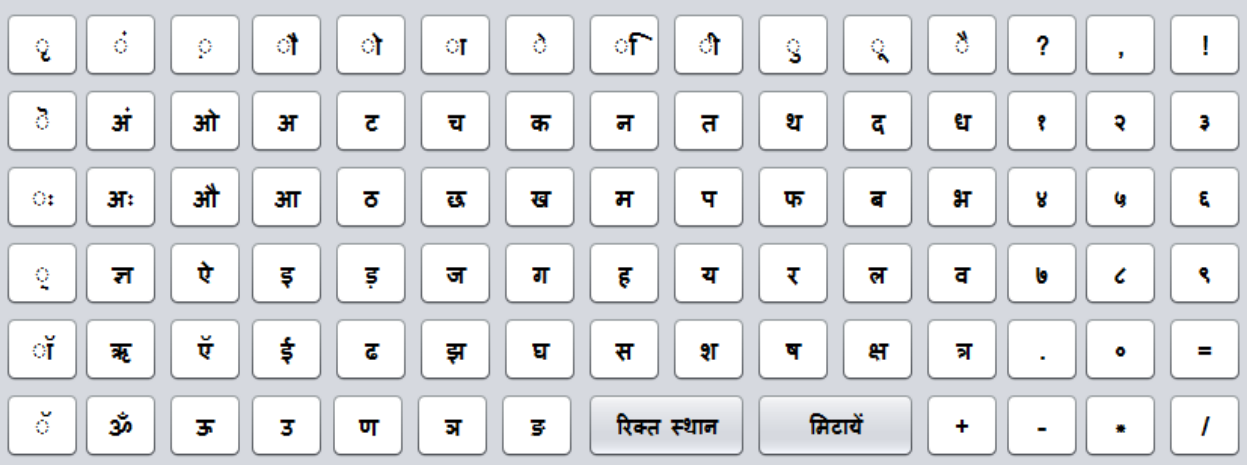

Figure 6. Virtual Keyboard of WRS

It is hard for unskilled operators and regular user to use standard keyboard for Hindi typing purpose, we have designed a virtual keyboard for those users. Virtual keyboard includes buttons for all alphabets and letters require for typing in Hindi language.

5.1 Data Results

A large set of training data help in producing better results. Information and resource for Hindi language is very less on the web as compared to of English language. We have used IIT Bombay Hindi corpus for this project [7]. Our file processing system found 2.852638 million words from 1175 different corpus files which we have processed to generate unigram, bigram and trigram. First we have parsed corpus files and arranged file data as one sentence per line in new files, then each line of the new files are processed to build unigram, bigram and trigram. WRS generated 136,432 unigrams, 989,775 bigrams and 1,270,052 trigrams.

5.1.1 Comparison of memory consumption:

Language model is designed with two data structure, hashmap and ternary tree. Both the data structures are well suitable for N-gram storage but hashmap has several advantages over ternary tree. Hashmap consume almost 60% of memory consumed by ternary tree and hashmap is two times faster than ternary tree

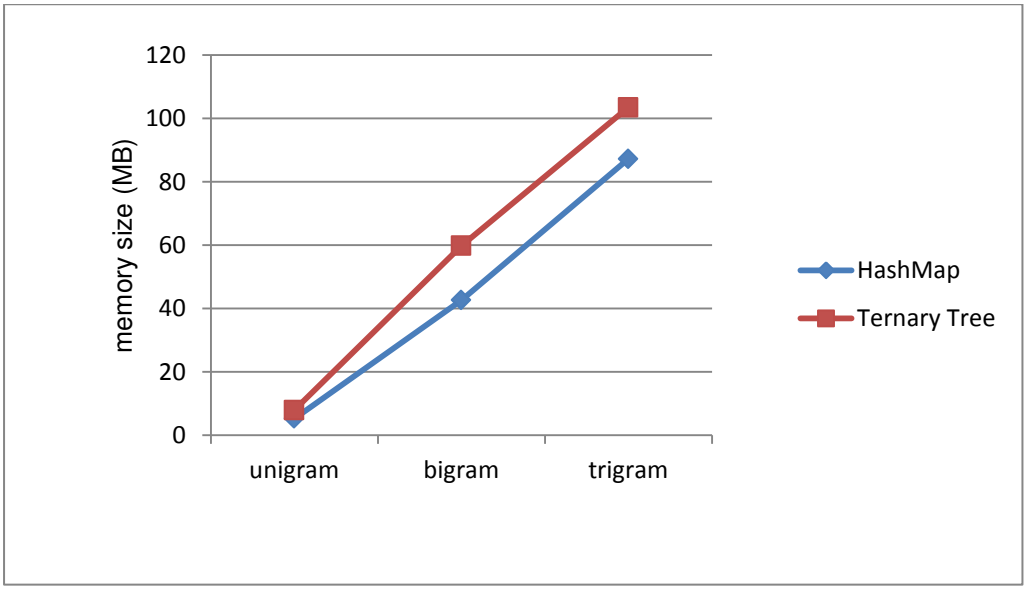

Figure 7. Comparison of memory consumption of Data structures for WRS

5.1.2 Comparison of searching time:

Searching speed is main concern in WRS hence response time for word prediction should be minimal. We have used two different data structures, one is based on tree and other is a hashmap. Ternary Tree is used to implement unigram, bigram and trigram sequence. Other than ternary tree, we have used hashmap and hashmap shows significant improvement over ternary tree implementation because searching time in hashmap is constant as compare to of ternary tree which take $log(n + k)$ time.

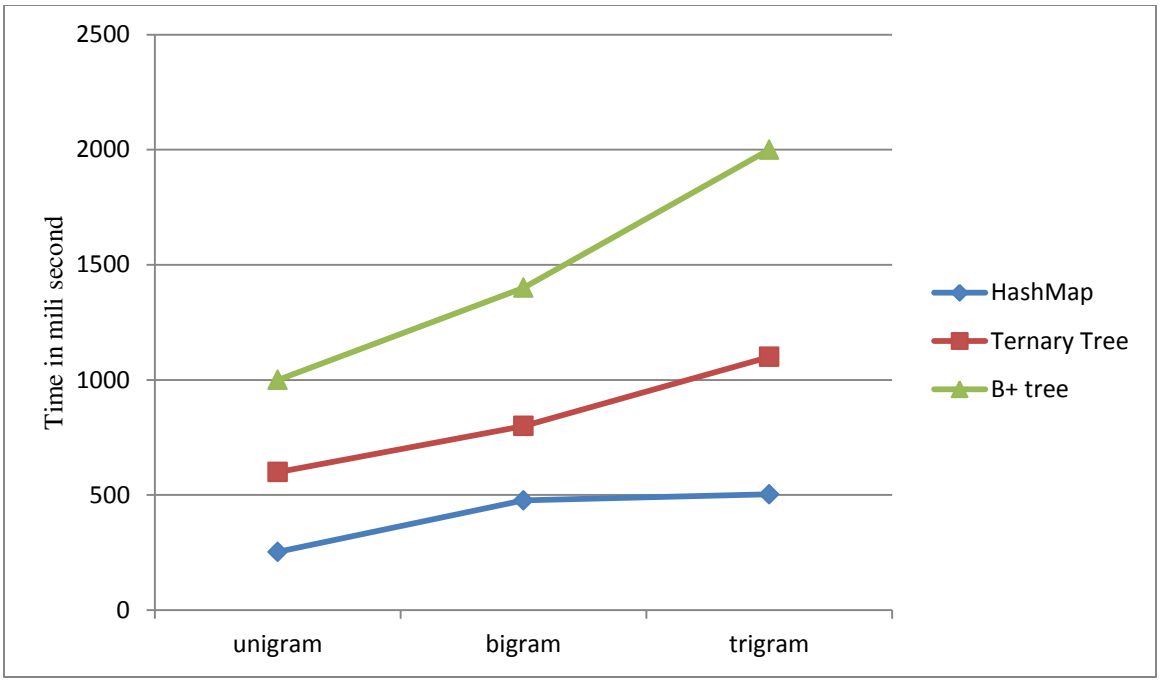

Figure 8. Comparison of searching time of data structure for WRS

5.2 Experimental Results

We have observed significant improvement in processing input words first by correcting spelling errors and then providing good suggestions to the user. It is observed that 87.33% of errors are corrected with the threshold value of 3, it means only those words which have edit distance less than threshold value will be corrected. Using WRS 92.67% of errors are avoided by selecting words from recommendation list. On the other hand 86.66% of words selected from the recommendation list were top 4 words of the list because it is important to have most likely words at high ranking. It reduces the total keystrokes needed to type complete word. Using WRS we have saved an average of 42.16% keystrokes per word by recommending most likely word when user type initial letters of the word. With the help of Part of Speech tags trigram we are able to recommend word list for unknown sequence of words also.

5.2.1 Comparison of different models with WRS:

These are some projects designed for Hindi language modeling, we have analyzed these models to design WRS and made some comparison on the basis of some important parameters which are required to compute the efficiency and accuracy of word recommendation system. Some models are not available practical experiments so we have used their research data and compared them with other models and WRS. We found that word prediction speech of WRS is faster than other

models by a little margin and with POS tags trigram we have improved the accuracy of prediction.

	without WRS	hIndiA	Lipika	WRS
Typing rate (wpm)	4.04	12.86	7.38	13.02
Error rate (%)	15	2.5	NA	2.5
Hit rate (%)	0	92.46	85.71	92.25
Keystroke saving (%)	0	43.05	32.43	42.16

Table 8. Comparison of WRS with other models

Working of WRS is illustrated in figure 5, GUI of WRS contain Swing components a text area, a list and buttons for alphabets. Text area allows user to type their content in it. There is a well defined function executed in back end for every user action, a function read every character typed by the user and according to that input system generates list of words. List component in right side is used to store the word list generated by the back end. List shows the recommended words in an ordered manner, the most likely word shows on the top of the recommendation list

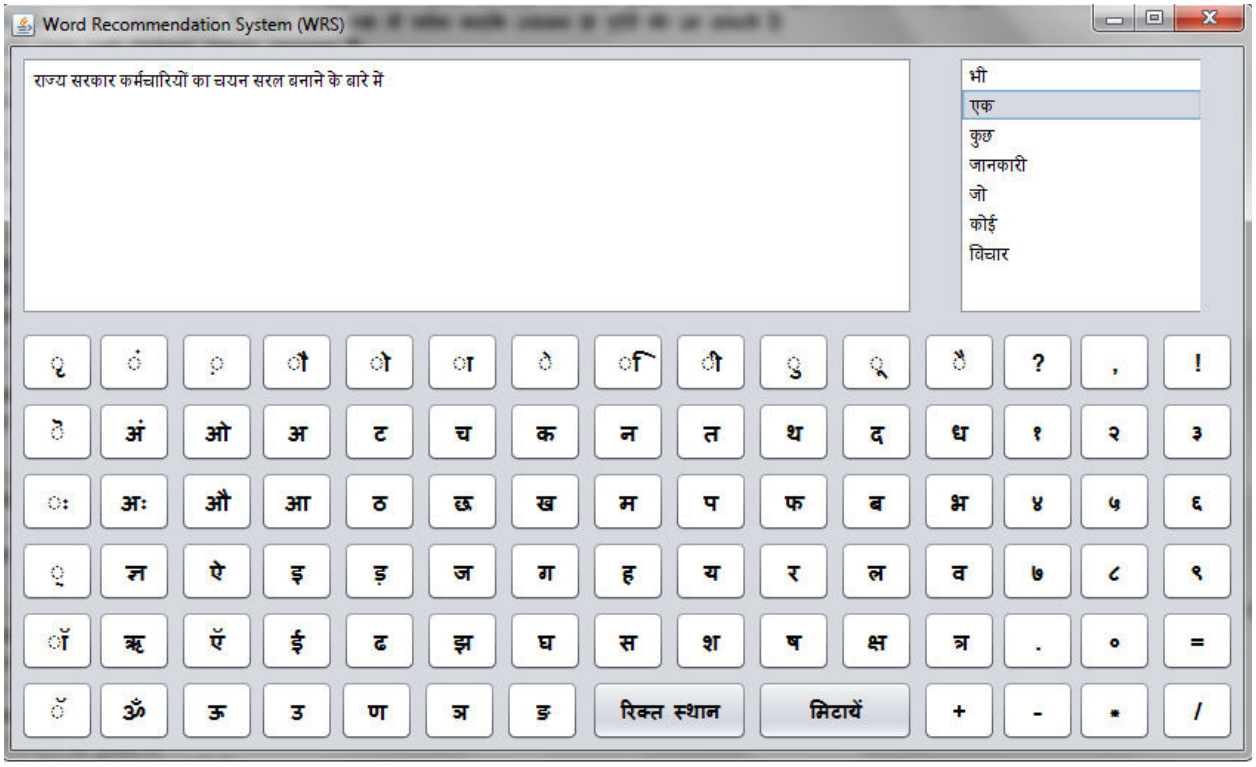

Figure 9. Working of WRS model

Based on our training data having almost 2.8 million words in 1175 different files, we have test the system many times and found that it does provide recommendation for all combination. Even if word does not belongs to training dataset. Here we are trying to give some input to the system, some of these words are known to system and some of these are unknown and does not occurred in training data. In either case WRS providing most likely next word for current word sequence.

Chapter 6

6 CONCLUSION AND FUTURE SCOPE

6.1 Conclusion

A good language model helps in many language processing and text processing applications like information retrieval, speech recognition, NLP and language translation. A language model could be statistical or syntactical; both have their own advantages and disadvantages.

We have proposed a language model which is a statistical model but it also uses Part of Speech (POS) tags trigram which helps in generating more accurate word list [6]. We have used hash map to build unigram, bigram and trigram which has several advantages. Firstly it provide O(1) searching time complexity which is faster than any other data structure [2], Secondly it saves memory because it does not require entry for word sequence which did not occurred in bigram and trigram. At the next step we have used POS tags trigram to deal with unknown word sequence. Spelling correction technique is used to normalize corpus data and user input, filtering input through spell checker reduce chances of error.

6.2 Future Scope

The WRS has a wide range of applications and usage in text processing area. There is some points listed below, which can be addressed further to get more accurate results in WRS,

- Normalization of corpus data in such a way that the language model can deal with confusing words.
- Word recommendations for miss spelled word could be more accurate if we can improve edit distance technique.
- Personalizing WRS for particular user would make it more accurate for that particular user.

REFERENCES

[1] Allison, B., David G., Louise G.: Another look at the data sparsity problem. Text, Speech and Dialogue. In: Springer Berlin Heidelberg. (2006).

[2] Balasubramaniam, L., Translation article knowledgebase - Spell it Right! (in the Context of Hindi). http://www.proz.com/translation-articles/articles/705/. (Jan 2015).

[3] Bharati, A., Rao, P., Sangal, R., Bendre, S. M.: Basic statistical analaysis of corpus and cross comparison. In: In Proceedings of ICON. (2002).

[4] Brown, P. F.: Class-based n-gram models of natural language. In: Computational linguistics 18.4, pp. 467-479. (1992).

[5] CDAC. Problems with existing unicode based engines. http://pune.cdac.in/html/gist/research-areas/set.aspx. (Oct 2014).

[6] Chen, K., Ming-Hong B.: Unknown word detection for Chinese by a corpus-based learning method. In: International Journal of Computational Linguistics and Chinese Language Processing 3.1, pp. 27-44. (1998).

[7] Chen, S. F., Joshua G.: An empirical study of smoothing techniques for language modeling. In: Proceedings of the 34th annual meeting on Association for Computational Linguistics, Association for Computational Linguistics, (1996).

[8] Consortium, U. 2011.: South Asian scripts-I. http://www.unicode.org/versions/Unicode5.0.0/ch09.pdf. (Oct 2014).

[9] Cormen, T. H., Leiserson, C. E., Stein, C., Rivest, R.: Introduction to algorithms. Vol 2, (2001).

[10] Dalal, A, Kumar, N, Sawant, U, Shelke, S.: Hindi part-of-speech tagging and chunking: A maximum entropy approach. In: Proceeding of the NLPAI Machine Learning Competition. (2006).

[11] Fazly, A.: The Use of Syntax in Word Completion Utilities, M.S. thesis, Department of Computer Science, University of Toronto. (2002).

[12] Ghosh, P. K., Knuth, D. E.: An approach to type design and text composition in Indian scripts. Ph.D. thesis, Stanford University. (1983).

[13] Goodrich, M. T., Roberto T.: Data structures and algorithms in Java. John Wiley & Sons. (2008).

[14] Google blog, http://googleresearch.blogspot.in/2006/08/all-our-n-gram-are-belong-to-you.html (Feb 2015).

[15] Guthrie, D., Mark H.: Storing the web in memory: Space efficient language models with constant time retrieval. In: Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics. (2010).

[16] Ishida, R.: An introduction to writing systems & unicode: A review of script characteristics affecting computer-based script support and unicode. http://people.w3.org/rishida/docs/unicode-tutorial. (Jan 2015).

[17] Isokoski, P.: Manual text input: Experiments, models, and systems. Ph.D. thesis, Department of Computer Sciences, University of Tampere. (2004).

[18] Joshi, A., Ganu, A., Chand, A., Parmar, V., Mathur, G.: Keylekh: a keyboard for text entry in indic scripts. In: CHI'04 extended abstracts on Human factors in computing systems, ACM, pp. 928-942. (2004).

[19] Koul, O. N.: Modern Hindi Grammar. Dunwoody Press, Hyattsville. (2008).

[20] Kowalski G.: Information Retrieval Architecture and Algorithms. (2011).

[21] MacKenzie, I. S. and Tanaka-Ishii, K.: Text Entry Systems: Mobility, Accessibility, Universality. Morgan Kaufmann, San Francisco, CA. (2007).

[22] Mohanan, T.: Argument Structure in Hindi. Center for the Study of Language and Information. Leland Stanford Junior University, CA. (1994).

[23] Pauls, A., Dan K.: Faster and smaller n-gram language models. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies. Association for Computational Linguistics. Volume 1. (2011).

[24] Robnek, D., Platos, J., & Snasel, V.: Efficient In-memory Data Structures for n-grams Indexing. In: DATESO, pp. 48-58. (2013).

[25] Sharma, M. K., Samanta, D.: Word prediction system for text entry in Hindi, In: ACM Transactions on Asian Language Information Processing, New York , USA, vol. 13 Issue 2, Article No. 8. (June 2014).

[26] Simon, C. P., Lawrence B.: Mathematics for economists. Vol. 7. New York: Norton. (1994).

[27] Stolcke, A.: SRILM-an extensible language modeling toolkit. In: INTERSPEECH. September (2002).

[28] Trnka, K., McCaw, J., Yarrington, D., McCoy, K. F., Pennington, C.: User interaction with word prediction: The effects of prediction quality. In: ACM Trans. Access. Comput. 1, 3, 1–34. (2009).

[29] Wandmacher, T.: AdaptiveWord Prediction and its Application in an Assistive Communication System. Ph.D. thesis, University of Tubingan. (2009).

[30] Zhai, C., John L.: A study of smoothing methods for language models applied to information retrieval. In: ACM Transactions on Information Systems (TOIS) 22.2: pp. 179-214. (2004).

LIST OF PUBLICATIONS

Premnarayan Yadav, Karun Verma.: Adaptive Word Recommender System with Fastest Data Structure for Devanagari Script. In: Eighth International Conference on Contemporary Computing (IC3), IC3-2015, (2015). [Communicated]

VIDEO LINK

https://www.youtube.com/watch?v=IvdPqh4FsXU&feature=youtu.be