An Efficient Approach to Keystroke Saving for the Blinds

Bharat Kapse
G. H. Raisoni College of Engineering (MS), India.

Urmila Shrawankar
G. H. Raisoni College of Engineering (MS), India.

ABSTRACT

Today many computer applications are available for the Blinds to interact with computer systems. Although computer interaction through keyboard is time consuming for visually impaired, their efforts can be minimized. Keystroke minimization or Keystroke saving is one of the approaches in minimizing the efforts of Blinds. The paper describes the work to achieve Keystroke saving. As the word prediction requires large database, in this work set of domain specific databases are constructed, where each domain database contains thousands of most commonly used words of that domain. It also construct prefix tree dynamically by modifying the Trie data structure. This dynamic prefix tree structure is used to perform prefix matching. The prefix matching is then analyzed to predict the required words from several domain specific databases used in this work. The paper describes the implementation and working of prefix matching and word prediction. The work presented in the paper is particularly useful for the blinds, as the work has considered all the difficulties of Blinds in interaction to computer through keyboard.The results of word prediction using modified trie are improved than trie based implementation.

General Terms

Human Computer Interaction, Data Structure.

Keywords

Keystroke saving, Word prediction, prefix matching, Trie, Domain database

1. INTRODUCTION

Today, the increasing pace in technology is fulfilling the needs of all users, including the Blinds. Over the years several researches have been carried out by Human Computer Interaction to develop the applications for the Blinds [4]. A portable e-book reader for the blinds is one of them given by Ramiro Vel'azquez [2].

This paper explains the working and implementation of Keystroke minimization system. The input to the system is prefix value entered by Blind users. Braille keyboard [1] is the important contribution for the blinds to interact with computer system. For each entered letter, the system forms the prefix by appending each letter. When the minimum possible prefix value is formed, the system predicts all the possible words, corresponding to entered prefix value. The mechanism by which words are predicted is called as Word Prediction. Keystroke Minimization system achieve it by implementing Prefix matching mechanism.

Nowadays, an auto complete feature is found in most text editors, in a large variety of browsing GUIs, for example, in browsers, in the Microsoft Help suite, or when entering data into a web form. The auto complete minimizes the keystrokes by predicting the required words. Recently, auto completion has been integrated into a number of (web and desktop) search engines like Google. Most of the above mention applications are using longest common subsequence approach, described by Bergroth [6].

To measure the ideal performance of the system, it considers the interaction by perfect user that never makes typing mistakes and will select a word from the predictions as soon as it is synthesized. The work has also considered the issue of time consumption in the selection of second or third word when predicting multiple words.

Although the system considers an ideal user interaction, word predictive interface designing is a complicated task. One of the features of Auto complete is the Inline word replacement. This work does not implement inline mechanism because the system users are visually impaired.

The utilization of digital resources [3] has started when the Braille language was developed by Louis Braille for the Blinds. Braille language is the best medium for the visually impaired to get the content in readable form. The development of speech synthesizer [5] was the major contribution in text to speech conversion.

The next section (II) describes the comparative study of prefix matching techniques. System architecture of the project is described in Section-III. Section-IV describes the methodology used in the project work.

Implementation details are explained in section-V. Section VI discusses the results obtained in the project implementation. Project conclusion is mention in Section VII.

2. COMPARITIVE STUDY OF PREFIX MATCHING TECHNIQUES

Previous section has explained the need and importance of word prediction system, as well as given the brief introduction of the work.

This section briefly categorizes the prefix matching techniques which were reviewed in the previous paper "prefix matching for keystroke minimization using B+ tree" [17]. The section also provides the analysis of various prefix matching techniques.

2.1 Prefix Matching

Prefix is nothing but the initial substring of a word. When the prefix is matched the word associated with that prefix is considered to be the desirable word. All the string matching techniques are also applicable for prefix matching. Also there are some special data structures designed for the prefix matching. This section describes them one by one. The prefix matching can be done by following two approaches.

- Fixed Length Prefix Matching
- Variable Length Prefix Matching

2.1.1. *Fixed Length Prefix Matching*

In the fixed length prefix matching, the prefix length is kept fixed and then the matched is performed. This type of matching can be performed using indexing mechanism efficiently. The indexing mechanism also takes the help of Hash functions to generate the indexes. Many tree technologies come under fixed length prefix matching. The following are some commonly used approaches.

i. Rabin Karp Algorithm

Rabin-Karp string searching [7] algorithm calculates a numerical (hash) value for the pattern p, and for each m-character substring of text t. Then it compares the numerical values instead of comparing the actual symbols. If any match is found, it compares the pattern with the substring by naive approach. Otherwise it shifts to next substring of t to compare with p.

ii. Best Matching Prefix

In this approach hash values are used in the form of prefixes and hence hash tables are created. Binary search mechanism is used in searching the required prefix as the hash tables are sorted.

Adding or deleting a single prefix can change the BMP [8] values of a large number of markers, and hence updating the forwarding table are expensive in the scheme.

2.1.2. *Variable Length Prefix Matching*

In the variable length prefix matching, the prefix length is not fixed and the match is performed depending on the number of words are associated with that prefix. This type of matching can be performed using Tree based mechanism efficiently. The Tree techniques are quite efficient and allow us to vary the length of prefix dynamically.

Many tree technologies can be configured to perform variable length prefix matching. The following are some commonly used approaches

i. Leaf-attaching
ii. Binary Search Tree
iii. Arbitrary length string matching
iv. Dynamic m-way prefix tree
v. TRIE structure

i. Leaf-attaching [8] algorithm takes a prefix tree as the input, and outputs a set of disjoint patterns. All the child (or non-leaf) patterns are merged with their parent (or leaf) patterns. Note that a pattern can be the child (prefix) of more than one parent pattern.

ii. In this approach, a memory efficient data structure based on a complete binary search tree (BST) [9] is presented. Each node in the BST includes a pattern and a match vector. With the corresponding BST built, the string matching is performed by traversing left or right, depending on the result of the comparison at each node.

iii. In this approach, the resulting prefix-patterns and suffix-patterns are merged into a split-pattern database, or dictionary. A prefix tree [10] is built for this database. The prefix tree is then leaf-attached and the leaf-patterns are extracted into a set of disjoint patterns.

iv. DMP tree [11] is a superset of B-tree. DMP tree does not support the repetition of pointers. To search all strings with the specified prefix, DMP tree applies all string search procedure. In the DMP tree all the prefixes of the data strings are at higher level. DMP tree support both the Longest Prefix Match [12] and Shortest Prefix Match approaches. DMP tree overcomes the worst case time complexity of Binary prefixes [13].

v. A Trie [15] is a data structure that allows strings with similar character prefixes to use the same prefix data and store only the tails as separate data. One character of the string is stored at each level of the tree, with the first character of the string stored at the root.

2.2. Analysis of Techniques

2.2.1. *Advantages of Fixed Length Prefix Matching*

- Same amount of time is required to predict any word.
- Searching is faster because of indexing and hashing mechanism.
- Fixed length prefix matching can be implemented with any tree mechanism.

2.2.2. *Limitations of Fixed Length Prefix Matching*

- Fixed amount of keystroke saving percentages, which are high compare to variable length prefix matching.
- Efficiency is reduced when too many words are available for the single prefix.
- Complexity increases when number of word increases.

2.2.3. *Advantages of Variable Length Prefix Matching*

- Words are predicted with high keystroke saving percentages.
- Variable length prefix matching provides flexibility in word prediction list.
- Updating of words does not increase the complexity.

In the prefix matching approaches variable length prefix matching is best suitable for word prediction. TRIE is the data structure designed to perform prefix matching by forming prefix tree [14]. TRIE has several advantages over other tree structures, particularly for variable length prefix matching.

With the advantages TRIE has some possible limitations.

2.2.4. Limitations of TRIE

- No count of words associated with prefix node.
- Extra searching over link of nodes.
- Missing linking to previous node.

The limitations of TRIE are the scope for further improvement in word prediction system. The modified TRIE structure which overcomes these limitations can be helpful to speedup word prediction, which will indirectly reduce the time required in word prediction.

3. SYSTEM ARCHITECTURE

The previous section has explained the need to develop keystroke minimization system for the Blinds. This section describes system architecture and also explains the functioning of each mechanism used in the project work.

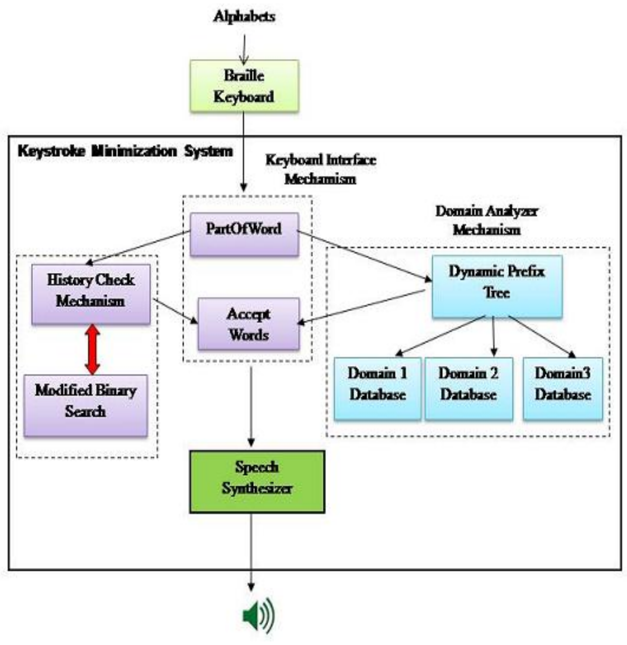

Fig 1: System Architecture for Keystroke Minimization

The project work is mainly divided into two parts.

- Keystroke Minimization System
- History Checking Mechanism (HCM)
- Keystroke Minimization System

3.1.Keystroke Minimization System

The Keystroke Minimization System is implemented using Keyboard Interface Mechanism, History Checking mechanism and the Domain Analyzer mechanism.

- The Keyboard Interface Mechanism accepts prefix values from the Braille keyboard.
- The History checking mechanism predicts the word using history words.
- DA mechanism is used to select the domain specific database which is further used to build the prefix tree.

Keystroke Minimization system is integrated with Speech Synthesizer which synthesizes the predicted word by making the use of system sound. It is using the concept of dictionary adaptive speech driven interface.

Speech Synthesis is a mechanism which reads the text input and translates it into the audio output. By using the Speech Synthesizer user can listen any document like a speech. The implementation of architecture requires the use of methodologies. The next section describes the methodologies used in the system implementation.

The two important mechanisms implemented in this system is explained below.

3.1.1. History Checking Mechanism (HCM)

History Checking Mechanism (HCM) maintains and controls the history words used by the user in Keystroke minimization approach. The HCM not only keeps the history words but also contribute in word prediction by using efficient Binary search methodology [16]. HCM maintains the priority queue to store the words, where the priorities are assigned on the basis of frequency of occurrence of a particular word.

The words in the priority queue are updated as the size of queue is fixed. Queue is updated by replacing the words based on "Least recently used" policy. The History Checking Mechanism is based on the following criteria.

- If there is only one word for the entered letter then it is the predicted word.
- If there is more than one word for the single letter then return only one word with the criteria of most recently used and smallest in length. And put the other words in the wait queue,

The wait queue is priority queue which keep the most recently used words by considering the length of the word.

3.1.2. Domain Analyzer Mechanism (DA)

Domain Analyzer is a special design provides the guidelines to find a substring match related to specific domain. Domain analyzer not only provides the domain specific match but also, it speeds up the substring match by reducing the searching area.

4. METHODOLOGY USED

The limitations of best available methodology motivate the research work to improve the efficiency and performance of a system. If the limitations are overcome by modifying the existing structure with improved performance, then the research is successful.

In the analysis of methodologies TRIE is found to the best available methodology.

The limitations of TRIE methodology are the scope for further improvement in word prediction system. The modified TRIE structure which will overcome these limitations can be helpful to speedup word prediction, which will indirectly reduce the time required in word prediction. The project names this modified TRIE as Dynamic Prefix Tree mechanism.

The main research contribution in this project is the design of Dynamic Prefix Tree, which is a modified version of TRIE. It overcomes the limitations of TRIE and improves the Word prediction performance. The DPT is described as follows.

4.1 DPT based Prefix Matching

All the String matching techniques mentioned in section 2 or other available can be used as a prefix matching techniques. Here also the work implements most commonly used prefix matching technique by modifying according to the system requirement. This work implements prefix matching using modified "prefix tree", also give the name as "dynamic prefix tree" and compares it with the prefix matching using B+ tree [12] and prefix tree. This section explains the modified prefix tree structures as follows.

Dynamic Prefix Tree: Dynamic Prefix Tree (DPT) is shown in fig.3. DPT is a modified form of prefix tree. DPT is designed here especially for the word prediction for Braille users. But this modified structure can be used for other systems also. The only difference between Prefix tree and the BPT is the storage structure.

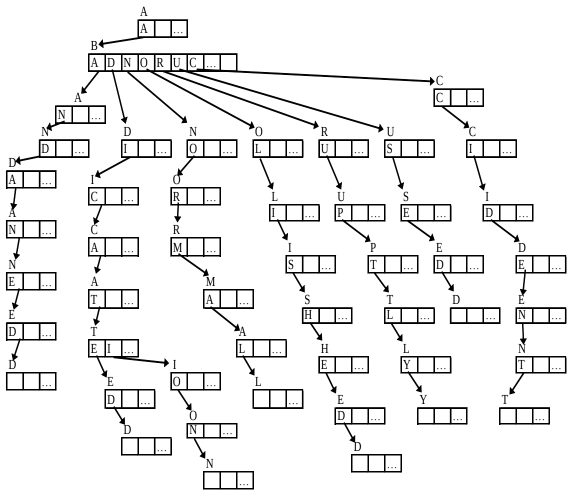

Fig 2: Dynamic Prefix Tree for the dataset shown in table 1.

In DPT links do not contain the character value of linked nodes in alphabetical order, where as it contains the link on first come first served basis.

DPT also contains the storages for the links but the node character value is stored in the link sequentially and not according to character position as shown in the fig.2.

Next section describes the implementation work of Keystroke minimization system

5. IMPLEMENTATION

The Keystroke minimization system is entirely focused on Blind users. The system achieves word prediction using prefix matching methodology. Prefix matching is implemented by optimizing the "prefix tree". Prefix tree is the most commonly used methodology in word completion.

The project constructs a database of more than 3000 words which are most commonly used. These words are retrieved from standard dictionary web sites. The prefix tree is modified to further reduce the time required for the word prediction. This work accepts minimum possible keys to predict the word.

This approach provide maximum of three options to select the predicted word. These maximum three words are chosen from the retrieved word through database by considering their word rating.

5.1 History Checking Mechanism

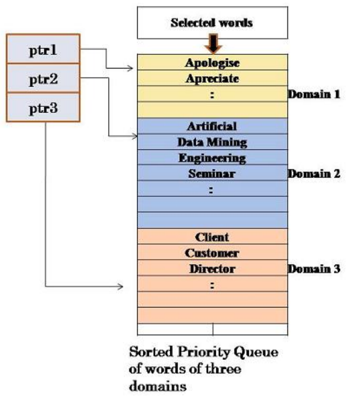

Fig 3: Priority Queue structure for HCM

The history checking mechanism is implemented in several small tasks, which are described as follows:

5.1.1. HCM range search

The HCM keep the recently predicted words in alphabetical order and there can be multiple words started with same letter. Hence range of predicted words is decided through the binary search, by using the following algorithm.

ALGORITHM 1. HCM Range Search

Input: The algorithm accepts values for low1, low2, high1, high2, mid.

Output: Most frequently used word which contains the entered prefix.

```
repeat until low1 is less than or equal to high1
if low1=high1,
high1=mid-1;
else,
if  low1>high1
low1= mid+1;
end
repeat until low2 is less than or equal to high2
if low2=high2,
high1=mid-1;
else,
 if  low2>high2
low2= mid-1;
end
```

The algorithm Range_Search is implemented to get the list of history words from history database. The algorithm range search implements nested binary search mechanism, where for every character match the search area becomes half of total area. As shown in the figure, algorithm first locates the range of all the words and then it searches for the prefix match in the found range.

When the range search is completed, we get the list of most predicted words. The list has a lower end and an upper end.

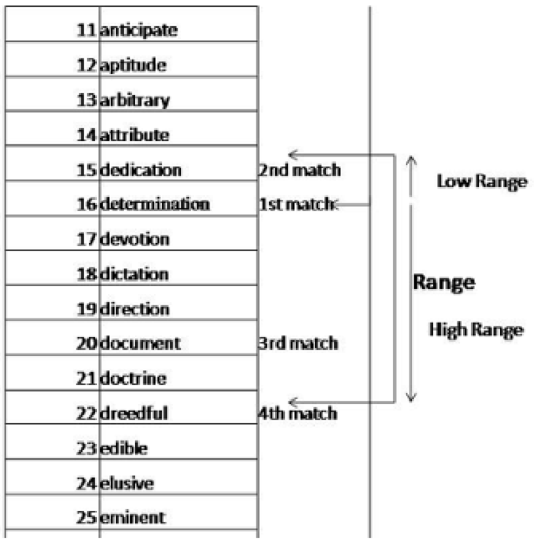

Fig 4: Working of HCM range search

5.1.2. *HCM Prefix Matching*

- When the user enters 1st letter of the prefix value, the letter position is identified in the priority queue as shown in fig 3. Here it is referred as first match.
- Then for the next (2nd) letter of the prefix (depending on the number of words for the particular prefix), range search is done which identify the words which are to be predicted. This range search identifies low and high region for the prefix matching.
- When user enters the 3rd prefix value, maximum of three words are selected out of the searched range by considering the priority of the word.
- Finally the selected three words are synthesized through speech synthesizer. The words are spoken one by one to the user in the form of choice.

In the History Checking Mechanism following parameters are used.

- HCM count: is the approximate value in history checking mechanism, calculated by counting total number of correct predicted words from a particular domain.
- Domain num: is a variable: contains the value 1 if the last predicted word is from domain 1. It contains value 2 if last predicted word is from domain 2 and same for 3.
- Session rating: decides which domain is currently in use. It is given from the title rating and maximum num of keywords used in particular domain.

5.2 Domain Analysis and Word Prediction

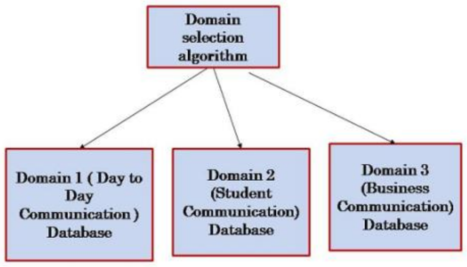
Domain selection algorithm
Domain 1 (Day to Day Communication) Database
Domain 2 (Student Communication) Database
Domain 3 (Business Communication) Database

Fig 5: Design of DA Mechanism

Domain Analyzer is the mechanism where the complete database is connected. The database is split into three domains. The role of domain analyzer is the selection of a particular domain to build the prefix tree. In Domain Analyzer some priorities are set for the selection of domain. Domain analyzer receives the feedback from history checking mechanism to select the proper database.

ALGORITHM 2. Word Prediction

```
Input: alphabet, tree root node, current node.
Output: list of predicted words.

1. repeat until all links of the current node are checked.
     if user letter=letter of current link,
       link to subtree;
       go to step 2;
     else,
       move to next link;
     end
2. if sub tree count>3
     wait for next user letter;
   else,
     retrieve all the words of sub tree with their word rating and store it into
     priority queue;
   else,
     retrieve top two words;
   end
```

Word prediction algorithm implementation require following sub tasks to be performed. These are:

- Words Retrieval
- Tree Building
- Prefix Matching
- Word Pronunciation

5.2.1. *Words Retrieval*

When the Braille user enter the first letter of the word, this letter is given to getWord() function. The function getWord() retrieve all those words from database which are started with the user entered letter. These words are stored in the "word log file". This word log file is then further use for the tree building.

5.2.2. Tree Building

In this project we are modifying prefix tree to build at run time and also to save unnecessary comparisons. This dynamic prefix tree is built when the first letter is entered by the user. Dynamic prefix tree overcome the problem of unnecessary comparison after prefix match to extract the remaining characters.

In the prefix tree after the prefix matched is done, all the links of the current node of prefix are checked to retrieve the remaining characters of the different words. This checking is done 26 times for each node. This overwork is reduced by dynamic prefix tree.

In the dynamic prefix tree, we are keeping 26 links and one separator. The separator will move from left to right depending on the number of entries.

It is observed that, for the four character prefix, very few numbers of words are predicted. This number may go up to 7 or 8. In such situation if we provide random linking by storing the links as first come must get first position, then unnecessary checking of remaining links is avoided, as we know that before separators, all links are available.

5.2.3. Prefix Matching

Following is the procedure to form prefix tree:

i. Node Creation

In this step node is created for every character of the word. Each node has 10 link elements and one data element. The data element stores the character for which the node is created and the link element keeps the position for each upcoming node to be connected to the current node.

As we have total 10 links so the each pointer will point to 10 different nodes.

ii. Node Linking

This method runs for each next word to be connected in the prefix tree. When Build tree function gets the word, it first extracts all the characters from the word and then check each character for the match with the previous nodes available in the sub-tree tree. If available, the current character is skipped. If the same character is not available in the sub-tree then new node is created with the current character.

5.2.4. Word Pronunciation

Word pronunciation task is accomplished by customizing speech synthesizer. The speech synthesizers are most commonly used tools in text to speech converters. The project customizes Free TTS which is developed in java.

6. RESULTS & DISCUSSION

The work implements Word Prediction by following two ways.

6.1 Word Prediction using B+ Tree

The part of this work is presented in the previous paper “Word Prediction using B+ Tree for Braille Users” [18]. In the B+ tree based implementation we get the following results for each of the technique.

Table 1: Database Structure for word retrieval

Sr .No.	Word	Word Rating	Initial Prefix
1	ABANDON	5	A
2	ABANDONED	3	A
3	ABDICATED	2	A
4	ABDICATION	2	A
5	ABNORMAL	5	A
6	ABOLISH	5	A
7	ABOLISHED	3	A
8	ABRUPT	5	A
9	ABRUPTLY	4	A

The above table shows the database structure for the B+ tree. Num field shows the serial number of the word stored in the database. It is auto generated number. Word field contains all the commonly used words of length greater than or equal to 5 characters. Wr field shows the word rating of each word stored in the database.

At the time of word prediction, multiple words are arranged according to their wr and words with higher word rating are predicted first. Prefix field contains the name of first letter of the word. This field is necessary to create dynamic prefix tree. When the first letter is hit by the user, all the words of that letter are retrieved by the use of prefix field.

Indexing mechanism [17] is used to speed up access to desired data. The simple example is the author catalogue in Library. When indexing need to be used in place of prefix value, to match the strings, unique index generation is really a challenging task. Here we present prefix indexing technique designed by using bitwise X-OR operation.

Function getIndex() initially performs the X-OR operation between two initial characters, then the result is shifted by 4 bit position using bitwise left shift.

```
Getting Word=bankrupt
After X-OR between 1st and 2nd character index value=3
After left shift of result index value=48
After X-OR between result and 3rd character index
value=94
After left shift of result index value=24064
After X-OR between result and 4th character index
value=24171
For the String: bankrupt,   index=24171
Getting Word=bankruptcy
After X-OR between 1st and 2nd character index value=3
After left shift of result index value=48
After X-OR between result and 3rd character index
value=94
After left shift of result index value=24064
After X-OR between result and 4th character index
value=24171
For the String: bankruptcy,   index=24171
```

Fig 6: Results of prefix index generation

Fig 6 shows the generation of prefix indexes. As discussed earlier, the X-OR operation is performed to generate unique indexes for unique prefix value.

The above indexes are generated by considering 4 character prefix value.

```
Entry[24171,]
Entry[24171,]
Entry[17010,24171,]
Entry[17010,24171,]
Entry[17010,24171,]
Entry[4469,17010,24171,]
Entry[4469,7273,17010,24171,]
Entry[4469,7273,17010,24171,]
Entry[4469,7273,7781,17010,24171,]
Entry[4469,7273,7781,17010,24171,]
Entry[4469,7273,7781,17010,24171,]
Entry[1140,4469,7273,7781,17010,24171,]
Entry[1140,4469,7273,7781,17010,24171,49011,]
Entry[1140,4469,7273,7781,17010,24171,42100,49011,]
Entry[1140,4469,7273,7781,17010,24171,42100,43363,4901
1,]
Entry[24171,]
          Entry[1140,4469,7273,7781,17010,]
          Entry[24171,42100,43363,49011,91489,]
```

Fig 7: Results of B+ tree generation

The process of B+ tree generation is shown in fig 7. The result shows the node structure. The brackets indicate the node size and comma separates the node keys. The single node can contain any number of keys. It depends on the amount of element to be stored in the B+ tree.

The fig. 8 shows the results of word prediction using B+ tree when the user enter the prefix value. From the user entered prefix, prefix index is generated. This generated index is searched for the match over the generated B+ tree.

```
Total Keys=40
Entered Prefix=acco
For the prefix acco index=17263
word=accomplish
word=accomplished
word=accomplishes
Time taken to predict the word= 125 ms
Total number of words found=3
Predicted Words are
word1=accomplish
word2=accomplished
word3=accomplishes
```

Fig 8: Results of word prediction using B+ tree

When the match is found, all the words from database which are linked to the matched index are retrieved. As the database table contains only 3 words for the entered prefix, total numbers of found words are also 3 and predicted words are also 3. In case where number of words found are more than 3, top 3 words are predicted based on their word rating stored in the database with respective words.

6.2 Word prediction using Dynamic Prefix tree

In the Dynamic Prefix Tree based implementation we get the following results for each of the technique.

The database structure of Dynamic Prefix Tree is same as shown in table.1.

```
b a n k r u p t
        t c y
  a r r e d
    r i e r
       r s
 b e a u t i f u l
  e l i e v e
       e d
  e n e f i c i a l
      i t
       t e d
  e t t e r
 b o o s t e d
  o t t l e n e c k
  o y c o t t
 b r e a c h
```

Fig 9: generated Dynamic prefix tree for day-to-day database structure

Fig.9 shows the dynamic prefix tree formed for the initial prefix ‘a’. The look of the dynamic prefix tree is same as that of prefix tree, the modification is only in the internal structure. If we take the example of prefix “barr”, there are three words associated with this prefix as per the above tree. These words are “barred”, “barrier” and “barriers”. The word “barred” with highest word rating is synthesized first.

```
ABANDON
ABANDONED
ABDICATED
ABDICATION
ABNORMAL
ABOLISH
ABOLISHED
ABRUPT
ABRUPTLY
ABUSE
ABUSED
ACCIDENT
ACCIDENTAL
ACCOMPLISH
```

Fig 10: log file contains retrieved words from database

Fig10 shows the list of words stored in the word log file. These are the words which are retrieved from the database for the prefix letter ‘a’. Dynamic prefix tree is formed by reading this file. The word log file is maintained to avoid the repetition of tree building for the same prefix.

```
Accomplished
Bankruptcy
Adjourned
```

Fig 11: log file of predicted words

Fig 11 shows the log file which contains all the predicted words synthesized for the Braille users. These words are then stored in the history log file so that history mechanism can help in prediction of most commonly used words.

```
acco
bank
adjo
```

Fig 12: log file of prefix values entered by user

Process log file contain all the prefix values entered by the Braille users. This file is used to calculate the keystroke saving and for the selection of domain.

```
Total Nodes=76
Entered Prefix=acco
Number of predicted words=3
For the prefix acco
word=accomplish
word=accomplished
word=accomplishes
Time taken to predict the word =5779276 ns
The word accomplished is spoken
User switches to other word
The word accomplish is spoken
User switches to other word
The word accomplishes is spoken
Selected word=accomplishes
```

Fig 13: word prediction results using dynamic prefix tree

Complete results of dynamic prefix tree are shown in fig. 13. It illustrates the total working process of word prediction system.

7. COMPARISION OF RESULTS

Fig.14 shows the graph with prediction time in mili seconds. B+ tree takes much time in generating prefix index and in matching the prefix index over B+ tree as compared to prefix and dynamic prefix tree.

Dynamic prefix tree is little faster than Trie as it is modified, which is discussed earlier. The number of nodes and number of word predictions are same in Trie and dynamic prefix tree, but the architecture is little modified.

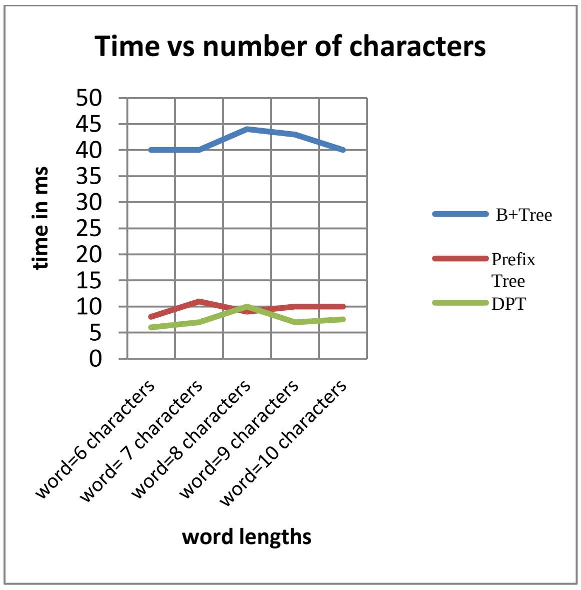

Fig 14: log file of Graph of comparison of different word prediction techniques.

8. CONCLUSION

- Keystroke Minimization System achieves word prediction in 3 to 4 keystrokes. That means whether the words are 8-10 characters long the user will take the effort to type only 3 to 4 characters. This clearly indicates that at least 50% efforts of keyboard users are minimized.
- This system saves on an average 3.5 keys behind every word for Braille users. Although some words are smaller in length, because of high percentage of keystroke saving, the work saves many keys on an average.
- On an average Keystroke saving percentage is between 50-55 %. As the system saves much keystrokes, the saving percentage are good with the consideration of length of prediction list. The saving percentage also analyzes the effort reduction. It also helps in calculating the time requirement in preparation of written documents. Although the percentage saving is not on higher side, but the system will maintain the performance after many user interaction. In comparison with the existing approaches length of prediction list is very much reduced. This indirectly save the user time selecting the predicted word.
- The database is built with the correct lexicon, which provides the user prediction of words with 100% correct spelling. This is particularly an additional benefit to the user as it needs not to worry about spelling mistakes. It has been seen over the years that many research publications or articles are rejected due to spellings mistakes. So the user interacting with computer through Word Prediction can help in getting correct words.
- The dynamic nature of tree generation helps in avoiding complexity issues of run time updations. This ultimately maintains the consistency of system performance, independent of size of lexicon. The second advantage of dynamic nature is the processing of database queries. Database queries can be fired after user finishes its interaction.

9. FUTURE SCOPE

Although Project results are improved, following parameters can be addressed in future study. These are,

- Bi-gram Lexicon
- N-gram Lexicon
- Corpus Study
- POS Study

If these issues are implemented & configured in the current system, better results may be obtained.

10. REFERENCES

[1] Guerreiro, T., Lagoa, P., Nicolau, H., Goncalves, D., and Jorge,J.A., “From Tapping to Touching: Making Touch Screens Accessible to Blind Users”, IEEE Multimedia, pp.48-50,October 2008.

[2] Ramiro Vel´azquez, Hermes Hern´andez, and Enrique Preza, “A Portable eBook Reader for the Blind”, 32nd Annual International Conference of the IEEE EMBS Buenos Aires, Argentina, August 31 - September 4, 2010.

[3] Pradeep Manohar, Aparajit Parthasarathy, “An Innovative Braille System Keyboard for the Visually

Impaired”,11th International Conference on Computer Modelling and Simulation,UKSim 2009.

[4] Bergroth, Lasse ; Hakonen, Harri ; Raita, Timo,"A survey of longest common subsequence algorithms", Seventh International Symposium on String Processing and Information Retrieval, SPIRE 2000.

[5] Coutinho, Luis Rodolfo Reboucas Girao, Anaxagoras Maia, Frota, Joao Batista Bezerra; Silva Jr., Elias Teodoro, “Device to documents”, 2012 Brazilian Symposium on Computing System Engineering.

[6] Shirbahadurkar, S. D. ; Bormane, Dattatraya S. ; Kazi, R. L, “Subjective and Spectrogram Analysis of Speech Synrhesisizer for Marathi TTS Using Concatenative Synthesis”, 2010 International Conference on Recent Trends in Information, Telecommunication and Computing (ITC).

[7] Robert Sedgewick and Kevin Wayne, “Algorithms 4th edition” , 2012.

[8] Y.-H. E. Yang and V. K. Prasanna, “Memory-efficient pipelined architecture for large-scale string matching”, In FCCM, 2009, pages 104 –111, april 2009.

[9] V. Srinivasan and G. Varghese, “Fast address lookups using controlled prefix expansion”, ACM Trans. Comput. Syst., 17:1–40,1999.

[10] Hoang Le, Viktor K. Prasanna, "A Memory-Efficient and Modular Approach for Large-Scale String Pattern Matching," IEEE Transactions on Computers, vol. 62, no. 5, pp. 844-857, May 2013

[11] Nasser Yazdani, Hossein Mohammadi, “DMP-tree: A dynamic M-way prefix tree data structure for strings matching”, Advances in Computing Systems Science and Engineering, Volume 36, Issue 5, September 2010, Pages 818–834.

[12] Bayer R, McCreight C, “Organization and maintenance of large ordered indexes”, Acta Inform 1972;1(3):173–89.

[13] Yazdani Nasser, Min Paul S, “Fast and salable schemes for IP lookup problem”, in: Proceedings of the IEEE conference high performance switching and routing, Heidelberg Germany; 2000.

[14] Black, Paul E, “"Trie" , Dictionary of Algorithms and Data Structures”, National Institute of Standards and Technology, Archived from the original on 2010-05-19.

[15] Yazdani Nasser, Min Paul S. “Prefix trees: new efficient data structures for matching strings of different lengths”, In Proceedings of the IEEE database engineering conference (IDEAS 01). Grenoble, France; July 2001.

[16] R ramakrishnan and J. Gehrke , “Database Management Systems”,3ed,2009.

[17] Urmila Shrawankar, Bharat Kapse, “ Prefix Matching for Keystroke Minimization using B+ Tree”, IEEE 8th International Conference on Computer Science & Education. Colombo, Sri Lanka, April 2013.

[18] Bharat Kapse, Urmila Shrawankar, “ Word Prediction using B+ Tree for Braille Users”, IEEE 2nd SCES, Allahabad, India, April 2013.