**“DYNAMIC TYPING TUTOR”**

***A***

***Project Report***

*submitted in partial fulfillment of the*

*requirements for the award of the degree of*

**BACHELOR OF TECHNOLOGY**

**In**

**COMPUTER SCIENCE & ENGINEERING**

**Specialization in**

**CCVT**

**by**

**Name Roll No.  
 Neha Mehta R110217096**

**Muskan Srivastava R110217091**

**Muskan Setia R110217090**

***under the guidance of***

**Dr. Rajeev Tiwari**

**Associate Professor**

**Department of Virtualisation**

****

**Department of Virtualization**

**School of Computer Science**

**University of Petroleum & Energy Studies**

**Bidholi, Via Prem Nagar, Dehradun, UK**

**July-Dec, 2019**

****

**CANDIDATE’S DECLARATION**

I/We hereby certify that the project work entitled **“Dynamic Typing Tutor”** in partial fulfilment of the requirements for the award of the Degree of BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING with specialization in CCVT and submitted to the Department of Virtualization at School of Computer Science, University of Petroleum & Energy Studies, Dehradun, is an authentic record of my/ our work carried out during a period from **August**, **2019** to **November**, **2019** under the supervision of **Dr. Rajeev Tiwari, Associate Professor, Department Of Virtualization**.

The matter presented in this project has not been submitted by me/ us for the award of any other degree of this or any other University.

**(Neha Mehta,Muskan Srivastava,Muskan Setia)**

**Roll No.-R110217096, R110217091, R110217090**

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

Date:19 November 2019 Project Guide

**Dr. Rajeev Tiwari**

Dr. Deepshikha Bhargava

Head – Department of Virtualization

School of Computer Science

University of Petroleum & Energy Studies

Dehradun – 248 001 (Uttarakhand)

**ACKNOWLEDGEMENT**

We wish to express our deep gratitude to our guide **Dr. Rajeev Tiwari**, for all advice, encouragement and constant support he has given us through out our project work. This work would not have been possible without his support and valuable suggestions.

We sincerely thank to our Head of the Department, **Dr. Deepshikha Bhargava**, for his great support in doing our project in **Area** at **SoCS**.

We are also grateful to **Dr. Manish Prateek, Professor and Director SoCS**, UPES for giving us the necessary facilities to carry out our project work successfully.

We would like to thank all our **friends** for their help and constructive criticism during our project work. Finally, we have no words to express our sincere gratitude to our **parents** who have shown us this world and for every support they have given us.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Neha Mehta** | **Muskan Srivastava** | **Muskan Setia** |  |
| **Roll No.** | **R110217096** | **R110217091** | **R110217090** |  |

**ABSTRACT**

Nowadays, everybody is interested in smart devices. However, most of them encounter difficulties when they have to type fast or have a tendency to make mistakes while typing. We analyse that text entry is essential for various applications and stands as one of the most difficult field to master.

In this project, we propose **Dynamic Typing Tutor**, an individual tutoring system for text entry that detects input stumbles and provides instructions. By conducting our study, we clarify common typing errors and most predicted typing style. Based on the study, we implement a Typing Tutor that will support learning text entry with accuracy and speed.

This project includes a simple implementation of the word predictor algorithm and typing error correction algorithm. So, accordingly, we will design a Typing Tutor that automatically detects input stumbles and provides instructions that help users to resolve input stumbles independently.

We define the word “input stumble” as an occasion when a user makes a mistake or does not know what/how to type next. We will then help the user to choose the next word with our tutor.

The different tasks performed by the dynamic typing tutor are:-

* Error Correction
* Word Prediction
* User Profiling

**TABLE OF CONTENTS**

**S.No. Contents Page No**

1. **Introduction 9**
   1. History 9
   2. Requirement Analysis 10
   3. Main Objective 11
   4. Pert Chart Legend 12

1. **System Analysis 13**
   1. Existing System 13
   2. Motivations 14
   3. Proposed System 15
   4. Modules 16

2.4.1. User Profiling 16

2.4.2. Error Detection And Correction 16

2.4.3. Word Prediction 16

1. **Design**

3.1 Data Flow Diagram 17

3.2 UseCase Diagram 18

3.3 ER Diagram 19

3.4 Flowchart 20

1. **Dynamic Typing Tutor**
   1. Tutor Overview 21
   2. Basic Information 21

4.2.1 Features 21

4.3. Programming the TYPING TUTOR 21

4.3.1. Quick Start Guide 21

1. **Implementation 22**

5.1 Algorithms

5.1.1 Edit Distance Algorithm 23

5.1.2 Trie(Prefix Tree) Algorithm 25

1. **Output screens 27**
2. **Limitations and Future Enhancements** 34
3. **Conclusion 34**

**References** 35

**LIST OF FIGURES**

**S.No. Figure Page No**

1. **Chapter 1**

Fig. 1.1 Pert Chart 12

1. **Chapter 3**

Fig. 3.1 Data Flow Diagram 17

Fig. 3.2 Use Case Diagram 18

Fig. 3.3 ER Diagram 19

Fig. 3.4 Flowchart 20

1. **Chapter 5**

Fig. 5.1 Trie (Prefix Tree) Algorithm 26

1. **Chapter 6**

Fig. 6.1 User Profiling(1) 27

Fig. 6.2 User Profiling(2) 28

Fig. 6.3 Error Correction 29

Fig. 6.4 Error Correction 30

Fig. 6.5 Word Prediction(1) 31

Fig. 6.6 Word Prediction(2) 32

Fig. 6.7 Word Prediction(3) 33

Fig. 6.8 Word Prediction(4) 34

1. **INTRODUCTION**

Smart devices like laptops offer new opportunities to improve the lives. To make full use of it, it is essential to master text entry. A Typing tutor is a simple application that we’ll be making in C to enhance the typing speed and accuracy.

This project includes a simple implementation of the word predictor algorithm and typing error correction algorithm. So, accordingly, we will design a Typing Tutor that automatically detects input stumbles and provides instructions that help users to resolve input stumbles independently. We define the word “input stumble” as an occasion when a user makes a mistake or does not know what/how to type next. In this project, we present the 3 steps to construct typing tutor.

The word predictor algorithm will be implemented using the concept of Trie(Prefix Tree) Algorithm, which will use the data of our case study. It is based on probability. We find probability of words and suggest accordingly which has the highest probability. The words are predicted from a already saved test file. We will also implement a code that will detect typing errors and will instruct the user to correct the typing mistakes with the help of a global dictionary.

**1.1 HISTORY**

In early 2000’s the increase in the use of laptops along with internet started. People

started using laptops with the internet facility for their office work and then slowly the use laptops increased to a very high number. Now, they are used worldwide by all types of people and for all types of work. It is used by children for playing games , for doing their homework. It is used by people who are working from home and many more.

Earlier, when the laptop’s craze was in the spreading phase , people did not had fast typing speed due to which they faced several problems like delay in their work ,typing mistakes etc.

After some years technology boomed and some error correction softwares were made that simplified people’s tasks, but still they were not sufficient.

As we talk about our era everyone is in very much hurry and mostly have to communicate through emails which means their typing speeds must be fast and errors should be minimum. To help people in such situations our dynamic typing tutor comes for rescue. Dynamic Typing Tutor helps the user by correcting the spelling mistakes and it predicts the next word that user can write which saves a lot of user’s time.

**1.2 Requirement Analysis**

The basic requirements of the customers are that they should have something that should help them with checking the spelling mistakes and also help them with their typing. Our project not only will help them to correct their typing mistakes but also with their typing speed by predicting the next word that the user wants to write. By this the user’s typing speed will increase along with the efficiency in typing.

So basically the requirements of the user are :-

1. Accuracy in typing

2. Fast Typing

3. Ease in typing

We have modeled the above requirements in our project so, that the users can have all what he desires. By using our Dynamic Typing Tutor , every user will save their time as they don’t have to recheck their work again and again for errors and also based on their previous files stored our tutor will help the user in writing the next word by prediction feature which is based on frequency, which means that in the previous files if the user has used that word many times then that word will be shown to the user first.

**1.3 Objectives**

1. To implement error-free word based Typing Tutor system.

2. Make it user specific application-Implement user profiling for improvements.

3. Embed it with prediction feature based on frequency.

**1.4 PERT Chart Legend**

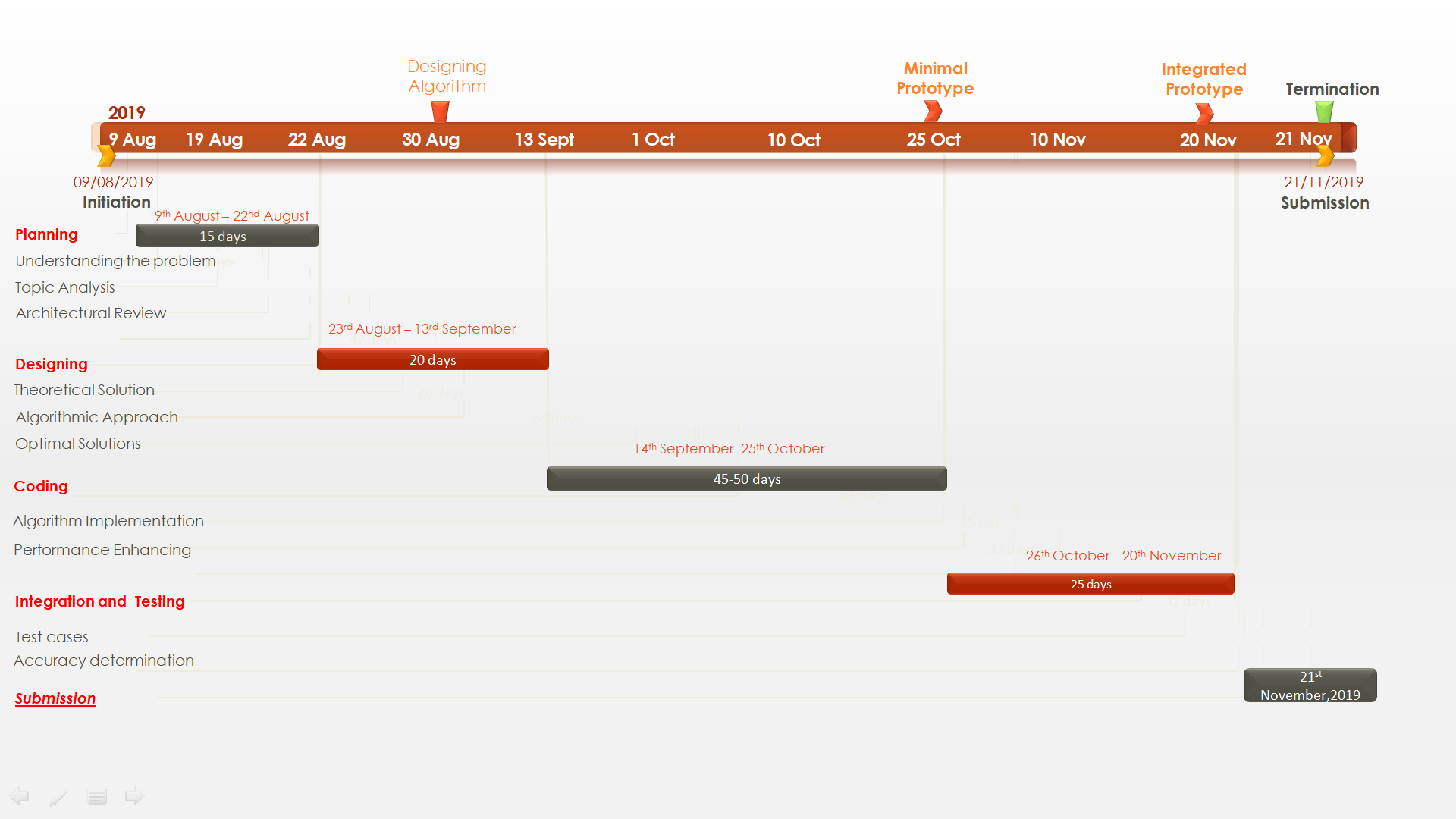
****

Fig 1.1

**2. SYSTEM ANALYSIS**

**2.1 Existing Systems**

Earlier, when the laptop’s craze was in the spreading phase , people did not had fast typing speed due to which they faced several problems like delay in their work, typing mistakes etc. After some years technology boomed and some error correction softwares were made that simplified people’s tasks, but still they were not sufficient.

As we talk about our era everyone is in very much hurry and mostly have to communicate through emails which means their typing speeds must be fast and errors should be minimum. Many systems are existing which solve the above problems like some error correction softwares and some tools that predict the next word but not with Trie prediction process.

By using Trie algorithm efficiency increases which is how our tutor is different and more efficient and user friendly. The next word predictions will be on the basis of frequency of the word used after the current word in previous sessions. To help people in such situations our dynamic typing tutor comes for rescue. Dynamic Typing Tutor helps the user by correcting the spelling mistakes and it predicts the next word that user can write which saves a lot of user’s time.

**2.2 Motivations**

People tend to make spelling mistakes and in this busy world typing should be fast enough. Sometimes while typing the word we want to type does not comes to our mind.

Given an input word and earlier user sessions stored, predict the next word that can occur after an input word and find and correct the spelling mistakes in the whole session , making the typing session efficient and user friendly.

We are proposing DYNAMIC TYPING TUTOR that will do the above task for us with maintaining efficiency and being user friendly.

**2.3 Proposed System**

The proposed system consists of 2 tasks. One is error detection and correction and the second is Word Prediction. With the help of the system , every user will save their time as they don’t have to recheck their work again and again for errors and also based on their previous files stored our tutor will help the user in writing the next word by prediction feature which is based on frequency, which means that in the previous files if the user has used that word many times then that word will be shown to the user first.

As the user will start typing the system will show the spelling mistakes and the similar words from the dictionary. Next, the tutor will show the words that have high frequency to be used next to the following word that is tutor will do the word prediction. By using both of the features of the proposed system the typing speed of user will become fast and they don’t have to spend more time on what word should they use after the following word.

**2.4 Modules**

**2.4.1 User Profiling**

As the user signs up a file named by his name is created, in which he can do his work and then all the data will be saved. So, as the user signs up user profiling takes place. User profiling will be helpful to predict the next word as the prediction will be on the basis of earlier user sessions stored.

User profiling made the prediction more efficient and user specific. The words predicted will be the ones that the user frequently use after the current word which will save his lot of time as he has to not spend time in thinking what to write next.

Two types of facilities are given to the user. First to signup if the user is new. Second is to login if the users not new. In the first case as the user enter a username and password a file corresponding to username is created and the user can work in that file. In the second case , user can login and then continue the work

**2.4.2 Error Correction**

Dynamic Typing Tutor will use the concept of Spell Checker to find the errors in spellings from the previously stored user sessions which will help the user to maintain an error free record.

A Spell Checker works by searching for a given string in its dictionaryof known strings. Any string not found in the dictionary is misspelling. The misspelling is alleged because it is a misspelling relative to the spell checker’s dictionary – which could be limited or incomplete. To correct a misspelling, the spell checker assumes that your misspelling must be a mutation of one of the strings in its dictionary. To suggest a correction, the spell checker searches its dictionary for strings **similar**to the misspelling. It then orders the potential corrections by their similarity and likelihood score. The string that is nearestto the misspelled word is suggested as a correction.

The sentences that user type are checked , if the words matches the word in the dictionary the word is accepted otherwise the nearest similar word is suggested to the user using edit distance algorithm.

**2.4.3 Word Prediction**

Dynamic Typing Tutor will use the Trie Prediction algorithm to predict the words that the user can use in the future, based on the previous sessions of the user.

Our typing tutor integrates a word prediction feature with the help of the Trie prediction model, that is, a mathematical probabilistic algorithm used to predict something on the basis of state attained in the previous attempt. This enables the user to produce an output which is basically a predicted words based on the prefix. The procedure starts by reading from the input and building a prefix table. Now, using the weightings based on the frequency of the word in the previous session of that user. The next step is to concatenate the string with the output that is generated by our system.

A Trie (Prefix Tree) Algorithm tells you the probability of hopping, or "transitioning," from one state to any other state. In the hands of meteorologists, ecologists, computer scientists, financial engineers and other people who need to model big phenomena, Trie (Prefix Tree) Algorithms can get to be quite large and powerful. For example, the algorithm Google uses to determine the order of search results, called PageRank, is a type of the Trie (Prefix Tree) Algorithm.

Thus, by using a Trie (Prefix Tree) Algorithm process, automatic typing tutor will check the last word that the user has entered and will find the probability of different words in the earlier user sessions that came after the last word user has entered and accordingly will show the suggestions to users.

1. **DESIGN**

**3.1 Data Flow Diagram**

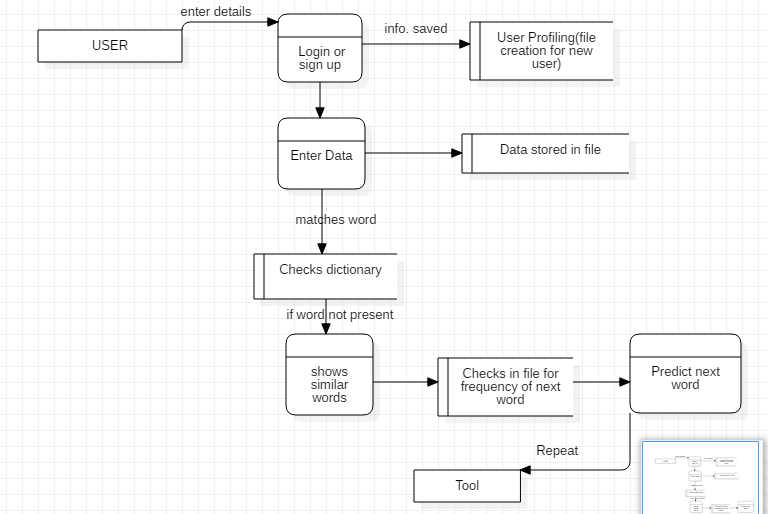
****

Fig 3.1: Data Flow Diagram of system

**3.2 Use Case Diagram**

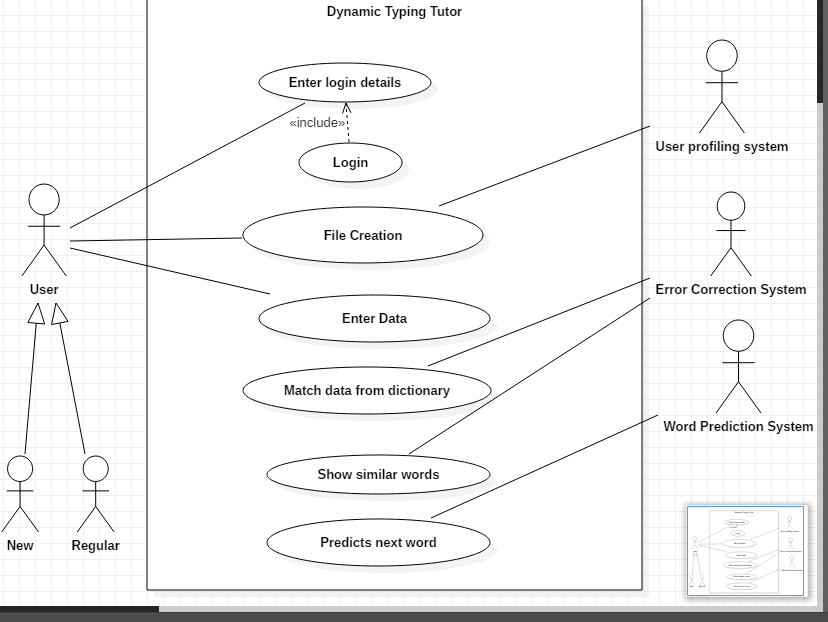
****

Fig 3.2: Use Case Diagram of system

* 1. **ER Diagram**

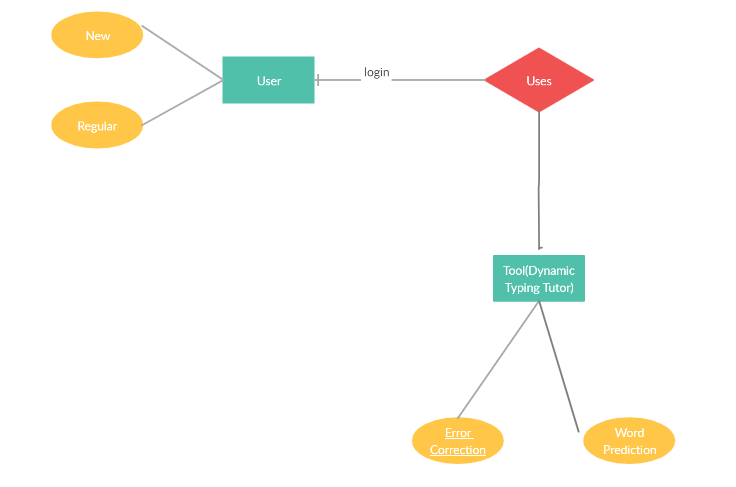
****

Fig 3.3: ER Diagram of system

* 1. **Flowchart**

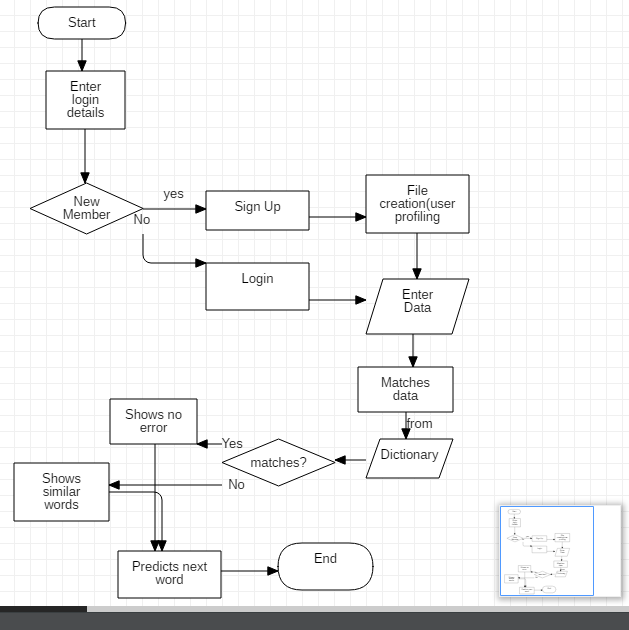
****

Fig 3.4: Flowchart of system

**4. DYNAMIC TYPING TUTOR**

**4.1 TUTOR OVERVIEW**

Dynamic Typing Tutor is a tutoring system that will store multiple user sessions and help users to detect spelling errors and help them to predict the next word with the help of earlier user sessions stored.

**4.2 BASIC INFORMATION**

**4.2.1 FEATURES**

The following are the features;

a. User profiling.

b. Error correction

c. Text prediction

**4.2PROGRAMMING THE TYPING TUTOR**

**4.2.1 A QUICK GUIDE:**

For the error correction part we have implemented the Edit distance algorithm. The sentences that the user types are checked, if the words match with the words in dictionary, the word is accepted; otherwise the nearest similar word is suggested to the user.

It also gives the edit distance i.e. the letter(s) of the word which has been changed or from which nearest suggestion has been made.

For the error prediction part we have implemented an algorithm named Trie algorithm. It uses the concept of data structure and tries. On the basis of previous inputs it suggests the word which has maximum frequency of coming after the input word.

1. **IMPLEMENTATION**

Our project is to implement a dynamic typing tutor in a console styled format, that is, it will be just like a command-line typist. We are trying to come up with an automatic process such that the user can improve his typing speed and accuracy with the help of predictive text and correctly-spelled language. The project effectively utilizes user-defined functions and the basic concept of file handling. The overall process of our dynamic typing tutor is as follows:-

**Step-1 File processing mechanism**

When a new user logs in the typing session, a file of input text will be created and a string containing user details is stored. There can be N number of different users using our command-line typist. Hence, separate files will be created for each user. These files will be stored and accessed whenever needed. This data has to be cleaned in a well-structured format before they will act as a lookup for the purpose of prediction and correction. The file length is also calculated and its limit is predefined. All the above operations will be performed using file handling in C language.

**Step- 2 Implementation of Next-Word Prediction**

Our typing tutor integrates a word prediction feature with the help of the Trie (Prefix Tree) Algorithm model, that is, a mathematical probabilistic algorithm used to predict something on the basis of state attained in the previous attempt. This enables the user to produce an output which is basically a predicted random word based on the prefix. The procedure starts by reading from the input and building a prefix table. Now, using the weightings based on the frequency of the word in the previous session of that user. The next step is to concatenate the string with the output that is generated by our system.

**Step- 3 Implementation of error-free typing**

The next step will be text correction. This lookup will be done by checking and comparing strings in the files stored in previous sessions of that particular user. Next, the system will introduce nearest suggested corrections based on the similarity of the error word with a set of words in the user files and the English dictionary being used. For this we have implemented the edit distance algorithm.

**5.1 EDIT DISTANCE ALGORITHM**

It takes two strings s1 and s2 and performs the following operations. It finds the minimum number of edit operation that are required for the conversion.

1. Insert
2. Remove
3. Replace

The basic idea is to process all the characters present in both the string starting from either of the side. Suppose we traverse from right side; for every pair of character being traversed, there are 2 possibilities.

Let, m be length of s1 and

n be length be s2

1. If last letter of both the strings is same then we ignore it, and do the recursion from m-1 and n-1

2. Else we consider all operations on s1. We will perform all 3 operations recursively and find the minimum cost of all 3 and consider the minimum value.

* 1. Insert: Recur for m and n-1
  2. Remove: Recur for m-1 and n
  3. Replace: Recur for m-1 and n-1

Steps:

1. Set n=length of s1;

Set m=length of s2;

If n=0; return m;

If m=0; return n;

Construct a matrix of size m\*n;

2. Initialize row 1 with 0….n;

Initialize column 1 with 0.…m;

3. Traverse each character of s1 (i from 1 to n);

4. Traverse each character of s2 (j from 1 to m);

5. If s1[i]==s2[j]; cost will be 0;

Else: cost = 1;

6. Set cell a[i,j] of the matrix to the minimum of:

a. The cell immediately above plus 1: a[i-1,j] + 1.  
 b. The cell immediately to the left plus 1: a[i,j-1] + 1.  
 c. The cell diagonally above and to the left plus the cost: a[i-1,j-1] + cost.

7. After all iterations are complete, distance is found in the cell a[n,m];

**5.2 Trie(prefix tree ) Algorithm**

Now to understand Trie or prefix tree model; suppose we want to develop a system which predicts the next word when we input an incomplete sentence. It treats every word as a state and predicts the next word based on previous state.

To execute it we enter probability distribution.

Let’s take an example:

1. I like dancing.

2. I like eating.

3. I love singing.

All the unique words from the given sentences are “I”, “like”, “dancing”, “eating” and “singing”; will form different states. In our case the probability distribution is from one word to another (which in mathematical terms means from one state to another).

In this case, there is a 100% chance that the 1st word will be ‘I’; as it always starts with “I”. For the second word, it will have to choose between words like “like” and “love”. Now, the task is to determine the probability that the next word will be “like” or “love” with the previous word “I”.

Approximately:

67% is the probability of “like”.

33% is the probability of “love”.

Again,

50% probability of dancing and 50% probability of eating to come after like.

And,

100% probability of singing to come after “love”.

Mathematical representation of above statements:

P(like | I) = 0.67

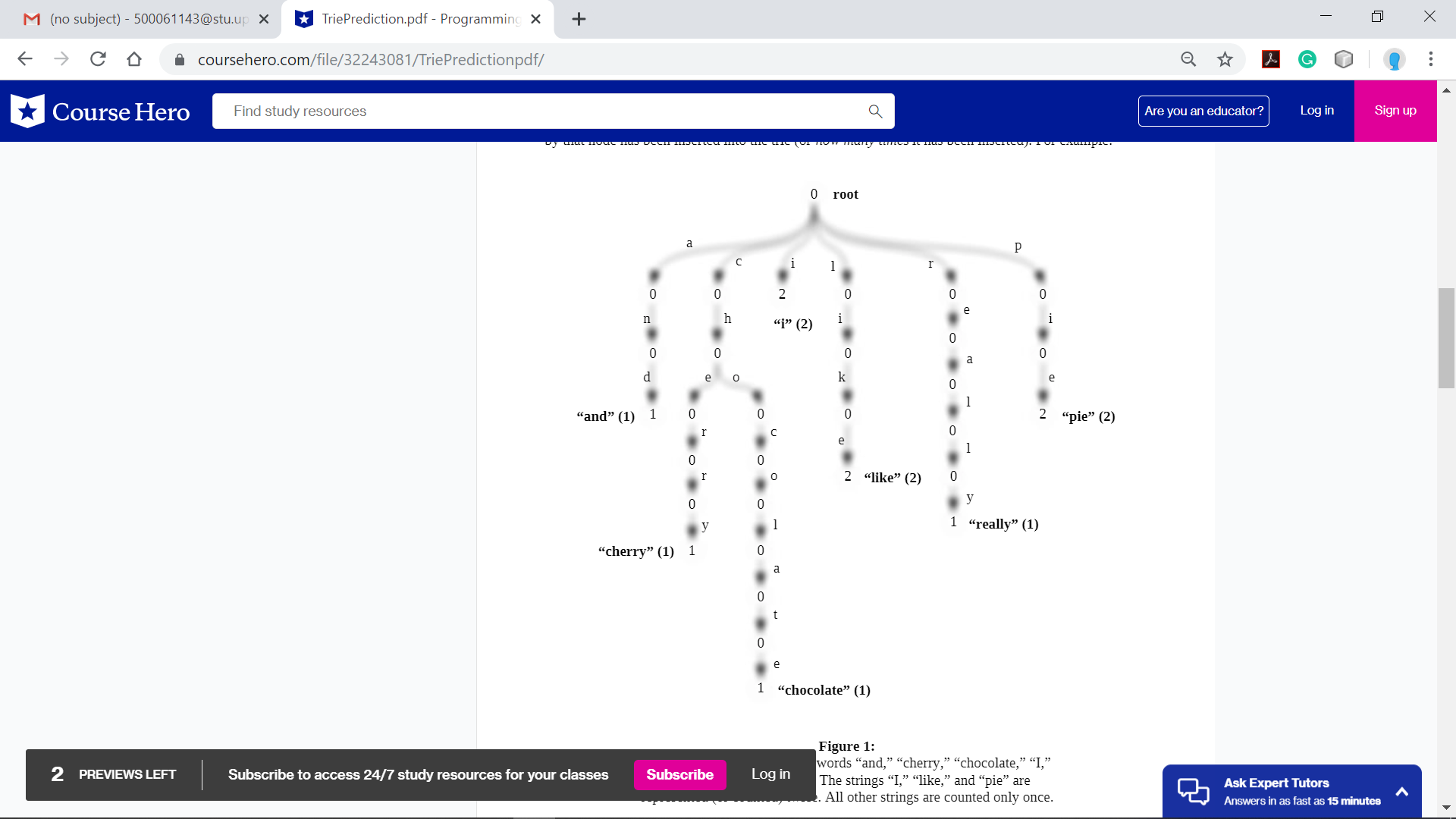
P(love | I) = 0.33

P(dancing | like) = P(eating | like) = 0.5

P(singing | love) = 1

The data structure called trie is used for the purpose of storing strings. The basic functionality of this data structure is insertion and look-up and the worst case scenario complexity is O(n). Here n represents string length.

The structure is simple as the nodes will not store the string, instead the edges will store the string. When we traverse the leaf node from the root node through the edges, we encounter the string stored. Also, the node contains a counter or a flag, which specifies the number of times the string that it shows has been inserted in the Trie.



**6. OUTPUT SCREENS**

**1. User Profiling**

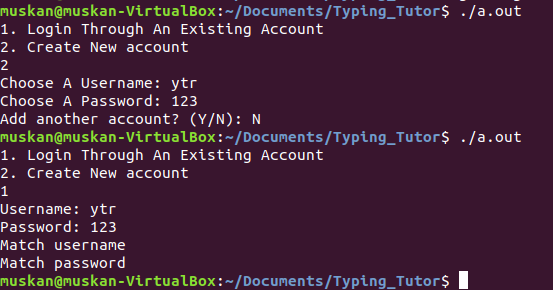
****

Fig 6.1

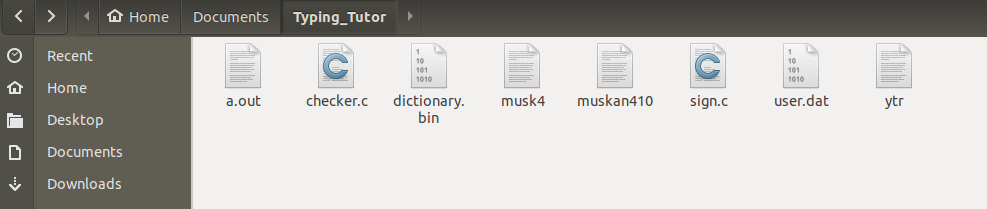
****

Fig 6.2

**2.Error Correction**

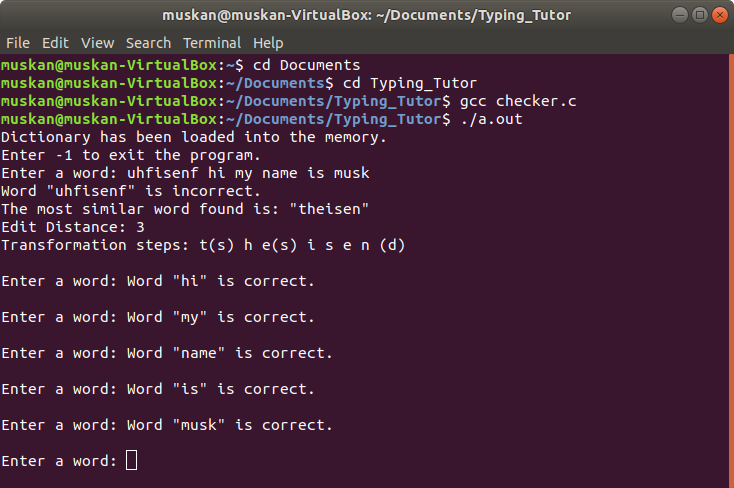
****

Fig 6.3

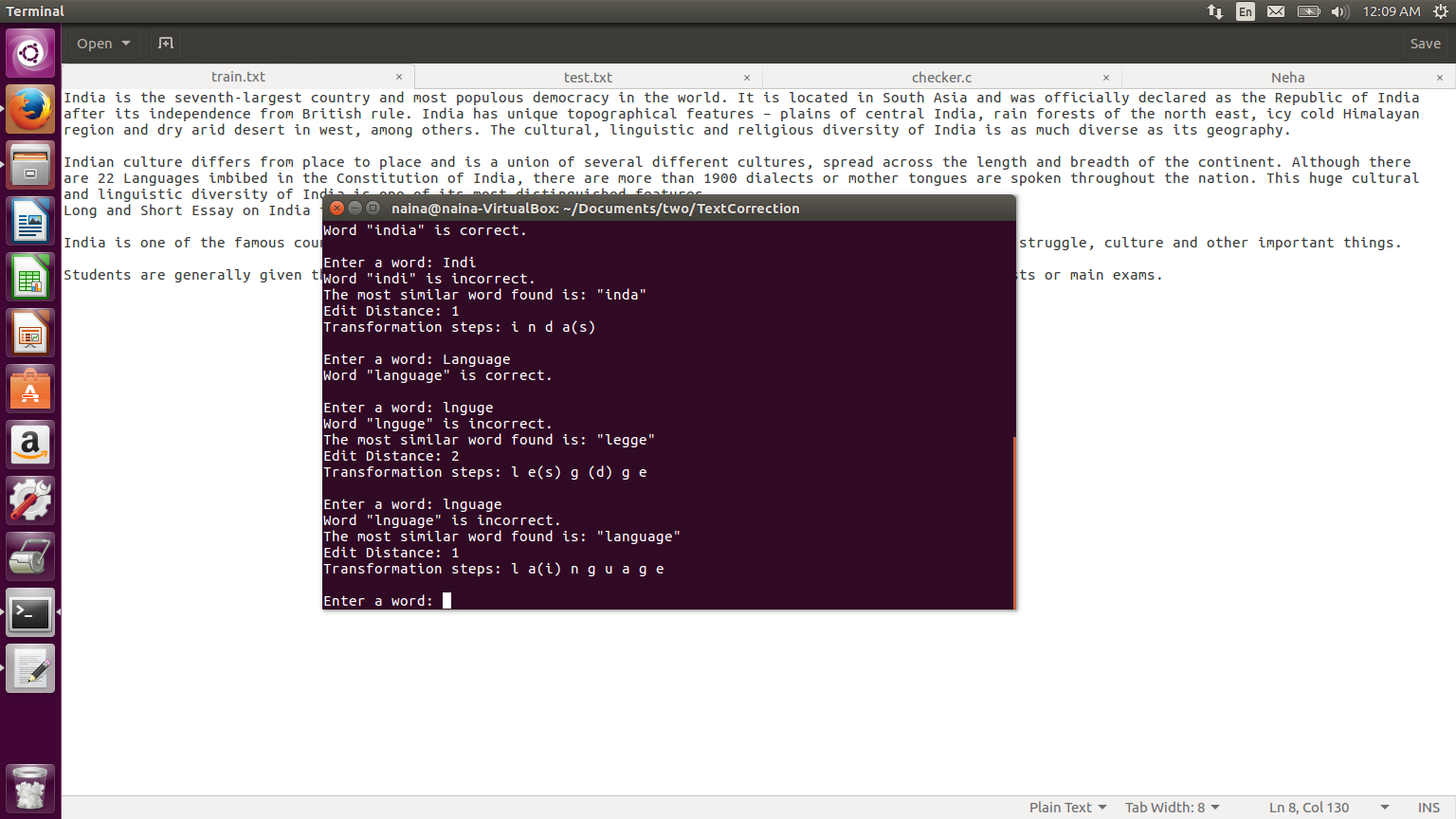


Fig 6.4

**3.Word Prediction**

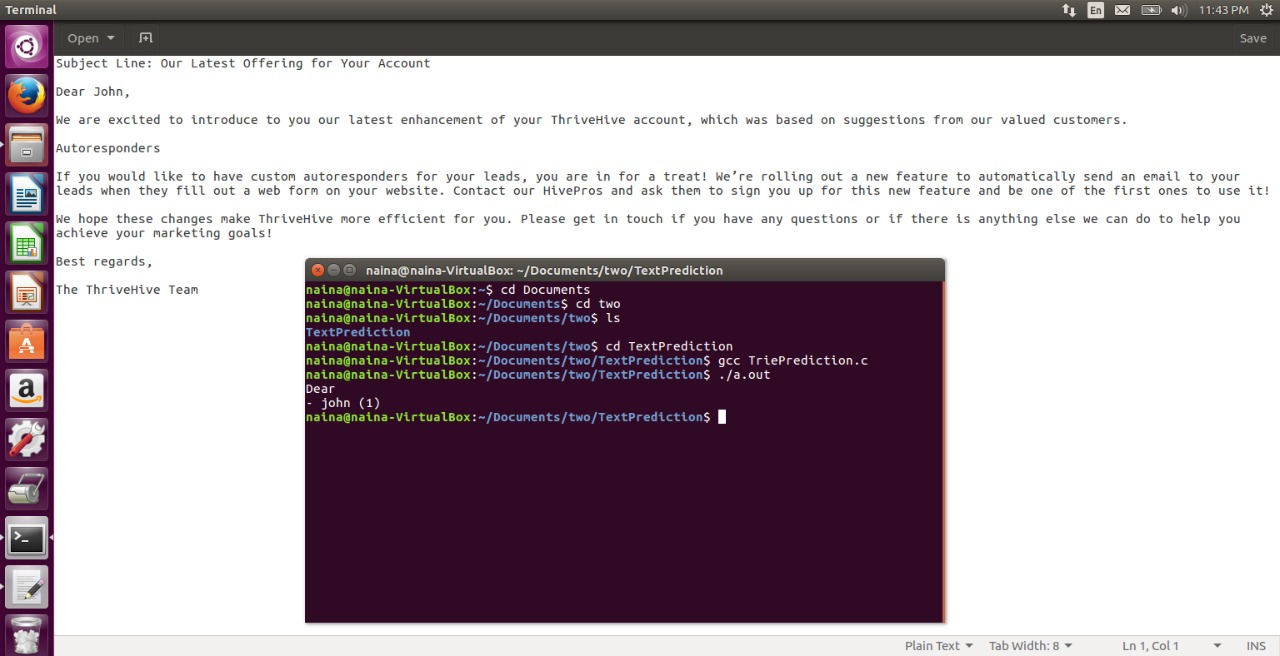
****

Fig 6.5

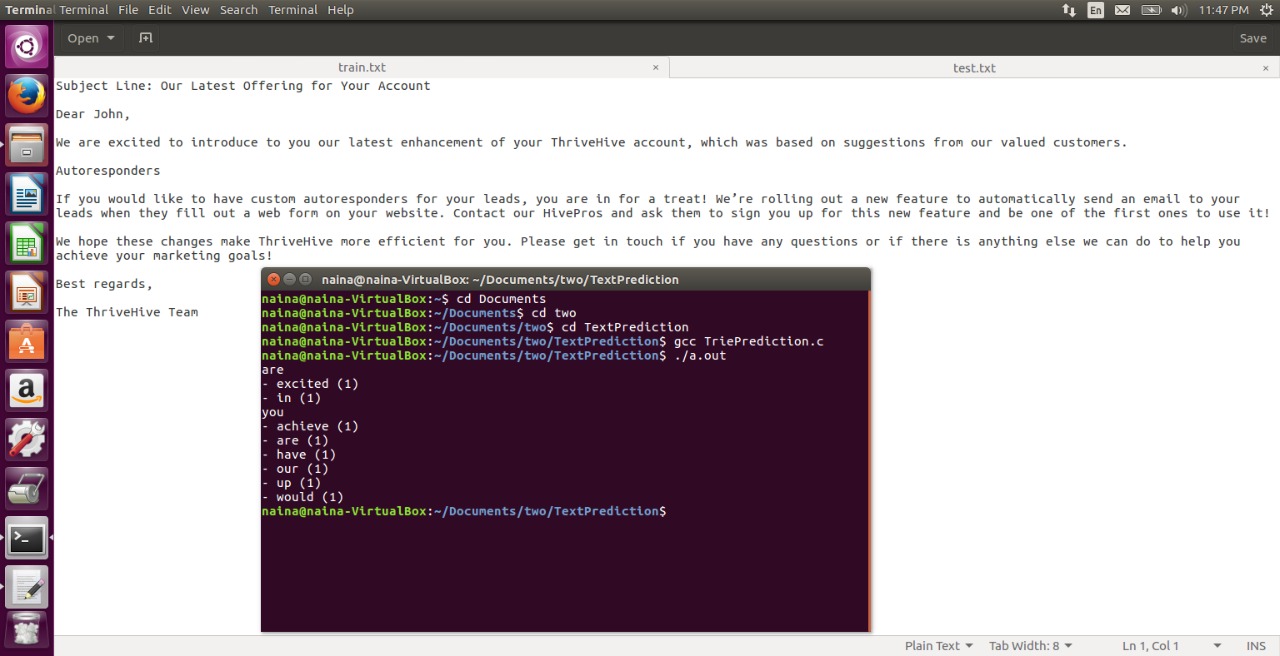
****

Fig 6.6

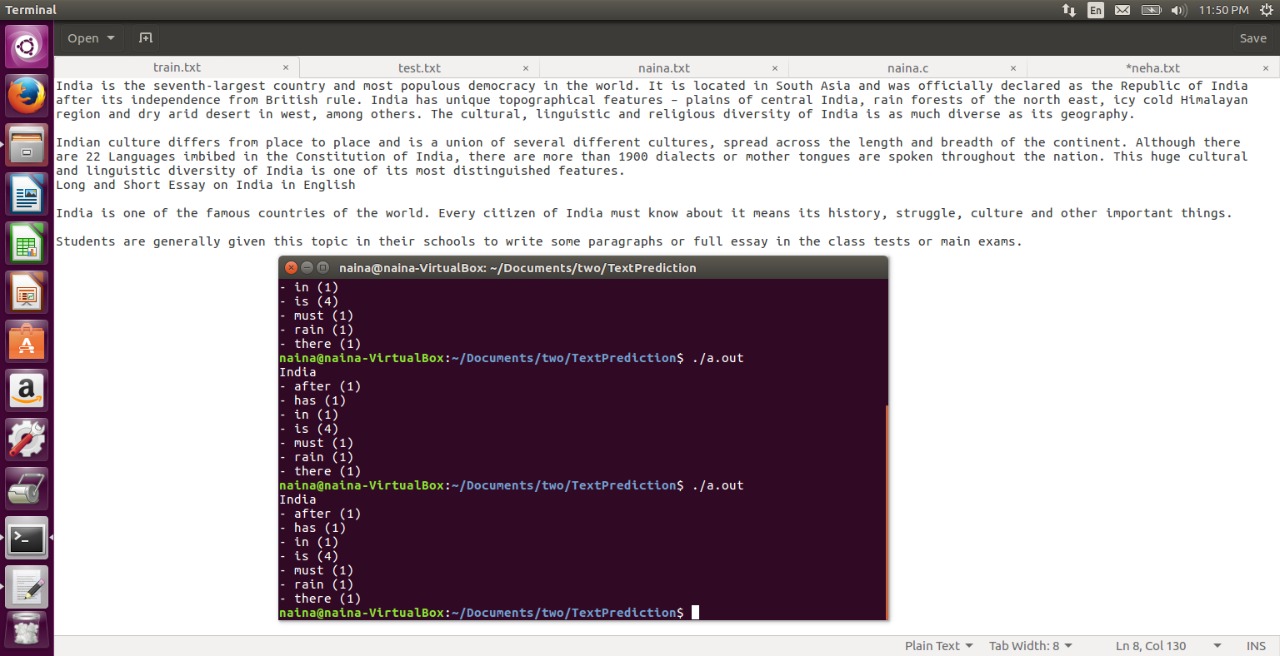
****

Fig 6.7

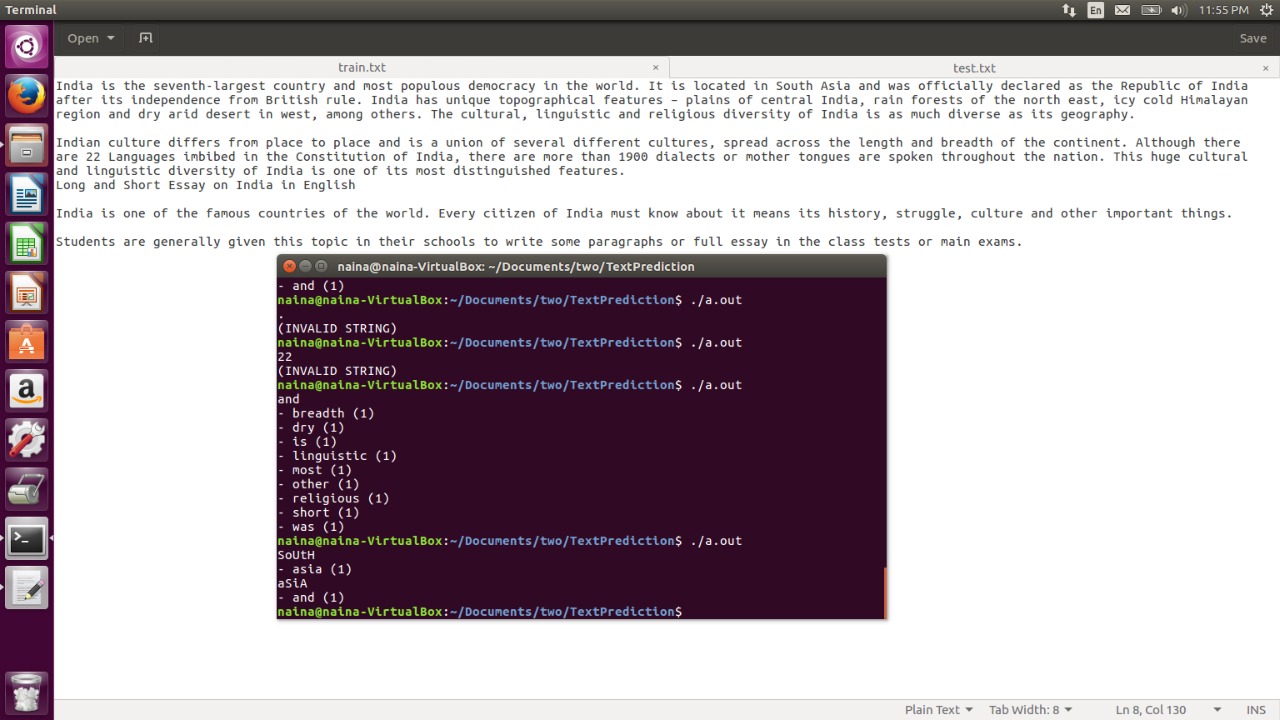
****

Fig 6.8

**7. LIMITATIONS AND FUTURE ENHANCEMENT**

The proposed solution- Dynamic Typing Tutor is a novel approach in the arena of text correction and prediction. However, we incur some limitations in the applied mechanism. First, the Trie or Prefix tree algorithm will demonstrate a slow performance when trained over a big corpus of words, thus there is a need to make enhancements in the algorithm.

In the future enhancements, work has to be done regarding the efficiency of this algorithm. Another alternate solution to this problem is Trie (Prefix Tree) Algorithms used in Natural language processing. Trie (Prefix Tree) Algorithm is another mathematical model that performs the same work i.e. next word prediction but makes use of advanced data structures such as hash table. The proposed mechanism works for next word prediction, but no development in the sentence generation. We would thereby like to move forward in this particular arena and implement the algorithm for generating more than one word or a complete sentence. Also, we can devise a methodology that takes into consideration more than one prefix i.e. bigrams or n-grams to predict the next word.

**7. CONCLUSION**

In the project- “Dynamic Typing Tutor”, we came across a past environment where users had to depend on proprietary error correction softwares. In this modern era, where everything is machine driven and open-source, dependency on computational systems has to be made flexible and according to user’s need.

Taking this in consideration, the project employs a word prediction and correction mechanism which is based on user entered data. We make use of two very popular algorithms known as the Editing distance algorithm and Trie Prediction Algorithm. The user enters our typing tutor by logging in or creating a new identity for himself. The tutor uses user profiling to save his credentials for the next time usage. The system makes use of a spellchecker to match the strings and therefore suggest correct word. Also, after some training text is fed to the Tutor, it can predict the occurrence of next word and also show the ranking of most desired word. Thus, making it quite easy to choose next word. The system works on various key parameters such as accuracy and speed by correcting and predicting words.

The implementation has faced some limitations and it woud be interesting to make use of other algorithms such as Markov Chain, word2vec, etc. and compare their results. Finally, we have tested the system over various cases.

**References**

[1] Hema P. H.\*, Sunitha C, *“Spell Checker for Non Word Error Detection: Survey”* in International Journal of Advanced Research in Computer Science and Software Engineering, Volume 5, Issue 3, March 2015, ISSN: 2277 128X

[2] Jane Cogie, Kim Strain and Sharon Lorinska, *"Avoiding the Proofreading Trap: The Value of the Error Correction Process" i*n The Writing Center Journal

[3] Kashansky, Vladislav & Kaftannikov, Igor. (2017). “*Application of TRIE data structure and corresponding associative algorithms for process optimization in GRID environment.”*

[4] Rene De La Briandais, *“File searching using variable length keys”,* Papers presented at the the March 3-5, 1959, western joint computer conference, p.295-298, March 03-05, 1959, San Francisco, California  [doi>10.1145/1457838.1457895]