NEXT WORD PREDICTION USING NEURAL NETWORKS

MINI PROJECT

Submitted by

GUNNU JAI RAJ	[RA2011003010171]
RAKSHANNA KANTHAN	[RA2011003010145]
ANUPAM PALAI	[RA2011003010206]

Under the Guidance of

MS.M. RAJALAKSHMI

In partial satisfaction of the requirements for the degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE ENGINEERING

SCHOOL OF COMPUTING

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR - 603203

MAY 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that the Mini project report titled **“ Next Word Prediction Using Neural Networks ”** is the bona-fide work of **RAKSHANNA KANTHAN - RA2011003010145, GUNNU JAI RAJ - RA2011003010171, ANUPAM PALAI - RA2011003010206** who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE	**SIGNATURE**
MS.M. RAJALAKSHMI	Dr. M. Pushpalatha
GUIDE	**HEAD OF THE DEPARTMENT**
Assistant Professor	Professor & Head
Department of Computing Technologies	Department of Computing Technologies

ABSTRACT

In this busy world, no one has time now. Next word prediction is a fundamental task in natural language processing, aimed at assisting users in generating coherent and fluent sentences. Neural networks have emerged as powerful models for next word prediction, capable of capturing complex language patterns and dependencies. This paper presents an overview of the current state of next word prediction using neural networks and explores its future scope.

The paper emphasizes the need for ethical considerations, including fairness, bias mitigation, and privacy preservation in next word prediction models. Lastly, it underscores the importance of continued research and innovation to advance the field and enhance the user experience across various applications.

Overall, this paper provides insights into the current state and future directions of next word prediction using neural networks, offering a foundation for further advancements in this field.

Table of Contents

LIST OF FIGURES

ABBREVIATIONS

NWP	Next word prediction
NLP	Natural Language Processing
AAC	Augmentative and Alternative Communication
WBS	Work Breakdown Structure
UML	Unified Machine Language

1. Introduction

1.1. Motivation

Autocomplete speeds up human-computer interactions when it correctly predicts the word a user intends to enter after only a few characters have been typed into a text input field. It's impact touches all aspects of our lives- from texting, to searching a result or file on the internet. For some people, this is what let's them do the most basic of tasks such as expressing themselves, communicating with others, searching and just doing their work easily. This project would make the entire process much more convenient for the user and they can save time and energy. Through our project, we truly wish to speed this process up by improving the efficiency and speed of word prediction and solve this particular issue for the people.

1.2. Aim of the proposed Work

To understand the dynamic data structure TRIE based on it develop a program having industrial application to predict words.

1.3. Objective(s) of the proposed work

➢ To understand the dynamic data structure tree used in developing the program.

➢ To understand the data structure 'trie' being used in the program.

➢ To construct a strong and efficient algorithm to develop the program which is editable and can be later used as a module for bigger software mechanisms.

To develop a real time program which is efficient and has fast processing and alsohas an industrial application

1. Literature Survey

1.1. Survey of the Existing Models/Work

Sr. No.	Title	Author	Algorithm proposed	Results obtained
1	Word and phrase prediction tool for English and Hindi language	Shashi Pal Singh, Ajai Kumar, Daya Chand Mandad, Yasha Jadwani	The tool is able to predict using multiple algorithms and they tried to build our own bilingual database which contains phrases of variable length along with the frequency of occurrence in corpus. Whenever a word or phrase is selected from the prediction list, its frequency isincreased by one. Consequently, the most frequently used word will have highest frequency/probability leading to increase in efficiency of predictions. Those words and phrases that are not present in	Word Prediction predicts list of word choices which the user may be willing to type after a word has been typed. Keystroke savings of various word prediction systems was found to be in

			the database but have been typed by the user are fed into the database for being used in future predictions. Thus, longer the tool is used, better results it will show.	between 37%-47% Conversion of text to speech as well as speech of text of these different languages can also lead to increase in usefulness of this tool. The tool performs predictions using three types of models i.e. frequency model, ngram model and recency model. Over all the application as well as the tool is highly easy to use for users having difficulty in forming words or typing text.

2	N-gram Word Prediction Language Models to Identify the Sequence of Article Blocks in English E-Newspapers	Deepa Nagalavi and M. Hanumanthappa	In this paper the link is established between different blocks of an articlewith the reading order of a sentence. It is identified with anN-Gram based linguistic processing approach for the retrieval of individual article from newspaper. The model predicts the preceding word knowing the previous content with the probability of a	It was challenging task to get 100% performance as it depends only on the training corpus of large data. However, to improve the
			word sequence. In this work an individual articleis identified efficiently with the reading order of the blocks.	word prediction task a linear interpolation model is used which combines trigram, bigram and unigram models. In this work single word prediction is conducted but a set of words or phrases can also be predicted to complete a sentence and merge the blocks of article with the matching sequence.

3	Pretraining Federated Text Modelsfor Next Word Prediction	Joel Stremmel and Arjun Singh	Federated learning is a decentralized approach for training models on distributed devices, by summarizing local changes and sending aggregate parameters from local models to the cloud rather than the data itself. They employ the idea of transfer learning to federated training for next word prediction (NWP) and conduct a number of experiments demonstrating enhancements to current baselines for which federated NWP models have been successful. The study compares federated training baselines from	The research offers effective, yet inexpensive, improvements to federated NWP and paves the way for more rigorous experimentation of transfer learning techniques for federated learning. For central pretraining with federated fine-tuning, we demonstrate

			randomly initialized models to various combinations of pretraining approaches including pretrained word embeddings and whole model pretraining followed by federated fine-tuning for NWP on a dataset of Stack Overflow posts.	a viable procedure but do not achieve performance greater than the federated training baseline with our large network. Simulating federated training conditions to pretrain word embeddings may also yield improved downstream performance by tailoring word representations to reflect different usage across non-IID datasets.

4	A Learning-Classificati on Based Approach for Word Prediction	Hisham Al-Mubaid	This paper presents a new word prediction approach based on context features and machine learning. The proposed method casts the problem as a learning-classification task by training word predictors with highly discriminating features selected by various feature selection techniques. The method is implemented and evaluated using several datasets and uses very small context (the preceding three words) to predict the followingword in that context with high accuracy.	The system achieved impressive results, compared with similar work; the accuracy in some experiments approaches 91% correct predictions. The method was evaluated extensively and compared with the Bayesian algorithm as a

				baseline. The experimental results showed that the approach can achieve impressive accuracy in percentages of correct predictions, which validates its efficiency.
5	A Classification Approachto Word Prediction	Yair Even-Zohar and Dan Roth	They present an approach to word prediction that is based on learning a representation for each word as a function of words and linguistics predicates in its context. To use an expressive representation of the context, they present a way that uses external knowledge to generate expressive context representations, along with a learning method capable of handling the large number of features generated this way that can, potentially, contribute to each prediction. Since the number of words "competing" for each predictionis large, there is a need to "focus the attention" on a smaller subset of these and thus, they exhibit the contribution of a "focus of attention" mechanism to the performance of the word predictor.	The success of this approach hinges on the combination ofusing a large set of expressive features along with a learning approach that can tolerate it and converges quickly despite the large dimensionality of the data. They believe that this approach would be useful for other disambiguation tasks in NLP. In the experimental study, approach presented is shown to yield

				significant improvements in word prediction tasks.
6	Advances in NLP appliedto Word Prediction	Synthema Srl	FastType, an innovative word prediction system that outclassestypical limitations of standard techniques when applied to in-flected languages. FastType is based on combined statistical and rule-base methods relying on robust open-domain language resources, that have been refined to improve Keystroke Saving. Word prediction is particularly useful to minimisekeystrokes for users with special needs, and to reduce misspellings for users having limited language proficiency.	FastType has been tried out and eval-uated in some test benchmarks, showing a relevant improve-ment in Keystroke Saving, which now reaches 51%, comparable to what achieved by word prediction methods for non-inflected languages. They additionally enriched the Language Model with morpho-syntactic information and provided the pre-dictionmethod with an on-the-fly Part-of- Speech word tagger and large lexicon dictionaries.

7	User interaction with word prediction:t he effects of prediction	Keith Trnka,John McCaw, Debra Yarrington, Kathleen F.	They present a study of two different word prediction methods compared against letter-by-letter entry at simulated AAC communicationrates.	They find that word prediction systems can in fact speed up communicatio
	quality	McCoy		n rate (an advanced system gave a 58.6%improve ment), and that a more accurate word prediction system can raise communicatio n rate higher than is explained by the additional accuracy of the system alone due to better utilization (93.6% utilization for advanced vs. 78.2% for basic)

1.4. Summary/Gaps identified in the Survey

From the survey, it became apparent that there's a need for simple, efficient and easy- to-use algorithm for word prediction. The required algorithm must be fast and should display output in a way which is easily readable. Our proposed algorithm caters to these exact needs.

2. Proposed System Requirements Analysis and Design

2.1. Introduction

In this segment, we discuss the requirements of our project and then its analysis and design in detail.

2.2. Requirement Analysis

2.2.1. Stakeholder Identification

Primary Stakeholders:

- Project leader
- Senior Management
- Project team members (algorithm management)
- Dataset Trainer
- Designer
- Resource Managers (database management)
- Product user group (Users)
- Project testers Secondary

Stakeholders:

- Project customer
- Consultants to the project

2.2.2. Functional Requirements

Our program is the main functional requirement where we will be using a programming language, example C, C++, etc. i. The system shall constantly access the dictionary file. User will give input in the form of a word (partial or full). Words matching the letters typed so far will be displayed on the screen. ii. If there are no matching words, then the system will ask if the user wants to add the word to dictionary iii. After every word prediction cycle completes, the system asks if the user wants to add more words. If yes, another cycle starts, else the system halts.

2.2.3 Non-Functional Requirements

a. Performance Requirements

- The system should be interactive to users.
- The interface is simple and easy to use.
- System is user friendly, self-explanatory and also it is provided with a help guide.
- This system can be used by everyone.
- Speed: The system should be made as fast as possible to reduce response time.
- Throughput: The throughput should be as high as possible. We should be able to attainmaximum output in minimum time.
- Resource Utilization: Resources are modified according to user requirements.

b. Safety Requirements

- Operation of weekly backups for the database should take place.
- The system should be tough and not prone to breakdowns.

c. Security Requirements

- The administrators maintain the system as per the maintenance contract.
- The system has to be secure from attacks.
- In case of breakdown should be stabilized soon.

d. Quality Attributes

a. Reliability

The solution should provide reliability to the user that the product will run with all the features mentioned in this document are available and executing perfectly. It should be tested and debugged completely. All exceptions should be well handled.

- Try to attain maximum reliability.
- Reliability will also be higher since we try to attain maximum accuracy.
- Maintain proper and updated dictionary files to improve reliability.

b. Accuracy

The product should be able to reach the desired level of accuracy. This prototype version is primarily for proving the concept of the project.

- The information provided in the dictionary files and by the user should be correct.
- Minimize the errors.
- All operations will be done correctly to increase the level of accuracy

2.2.3. System Requirements

2.2.3.1. H/W Requirements (details about Application-Specific Hardware)

The hardware requirements for the application are very low. The most important parts will be a fast-hard drive for the creation of a custom corpus. Microsoft Windows, *Nix, and Mac OS all have browsers that will work with the application. What follows are the minimum specifications for the various computers and other hardware:

I. Client Requirements

❑ Minimum CPU – P3/AMD Athlon 1.0 GHz+

❑ Minimum Disk Space – 512MB

❑ Minimum Memory – 256MB

❑ Minimum Network – 10/100MB Network

❑ Minimum Display – 800x600 Color CRT

❑ Required Software – Web Browser (Google Chrome)

II. Other Hardware Required

❑ Touch Screens/Keyboard

2.2.3.2. S/W Requirements (details about Application-Specific Software)

1. C++ compiler
2. Anaconda Distribution
3. NLTK Toolkit
4. Spacy toolkit
5. Windows Operating System/MacOS/Linux

2.2.4. Software Requirement Specification document

1. Introduction

Purpose

The goal of this document is to provide support information on Word Predictor (v1.0). It will attempt to explain the functionality of the program and the features it provides. Itwill, however, not be a user manual and provide instructions regarding the full workingof the program or the usage of the program.

Document Conventions

- Keywords in this SRS have been highlighted by making the text bold and italicized.
- Priorities for higher-level requirements can be considered to be inherited by the sub-requirements if priorities of sub-requirements are not explicitly mentioned.

Intended Audience and Reading Suggestions

This software specification document is mainly intended for 3 categories of users:

• Developers: Developers can review the project's capabilities and conveniently understand where their efforts should be directed to improve, refine and add features to it.

• Project Testers: They can use this document as the basis for their testing strategy since quite a few bugs are easier to find with the requirements document as reference..In this way, testing becomes more methodically organized.

• End Users: This document can also be read by the end users of this product who wish to read more in detail about the product along with its features and capabilit

- **Product Scope:**

The product aims to develop a next word prediction system using neural networks. This system will take in a sequence of words as input and predict the most likely next word that follows based on the context and patterns learned from a large corpus of text data.

Key Features:

1. Input Processing: The system will accept a sequence of words as input and preprocess them to remove noise, punctuation, and special characters. It will also handle capitalization and convert the text to a suitable format for further processing.

2. Neural Network Model: The system will employ a neural network architecture, such as a recurrent neural network (RNN), long short-term memory (LSTM), or transformer model, to learn patterns and dependencies in the text data. The model will be trained using a large dataset containing a diverse range of text sources.

3. Training Pipeline: The product will provide a training pipeline that allows users to train the neural network model using their own custom dataset or pre-trained models. It will include functionalities for data preprocessing, model configuration, hyperparameter tuning, and model training.

4. Prediction Engine: The trained model will power the prediction engine of the product. Given a sequence of words, the system will generate a ranked list of probable next words based on the learned patterns. The predictions will be based on the context and semantic meaning of the input sequence.

5. User Interface: The product will offer a user-friendly interface where users can input a partial sentence or phrase and receive real-time suggestions for the next word. The interface may also include options for adjusting prediction parameters, such as the number of suggested words or the prediction confidence threshold.

6. Customization Options: Users will have the flexibility to fine-tune the prediction system according to their specific needs. They can train the model on domain-specific data to improve prediction accuracy or adjust model parameters to balance between precision and computational resources.

.

.

Product Perspective:

The next word prediction product will serve as a tool to enhance the efficiency and productivity of users when typing or composing text. It aims to provide intelligent suggestions for the next word based on the context and patterns learned from a large corpus of text data.The product will be user-centric, focusing on delivering a seamless and intuitive user experience. It will integrate into various platforms and applications, allowing users to benefit from its predictive capabilities across different devices and environments.From a technical perspective, the product will leverage the power of neural networks to analyze and learn patterns in text data. It will utilize advanced natural language processing (NLP) techniques and machine learning algorithms to train a model capable of generating accurate and contextually relevant predictions.

The product will require a substantial amount of training data to build a robust language model. This data can be sourced from a wide range of textual resources, including books, articles, websites, and other relevant text collections. The system will employ data preprocessing techniques to clean and normalize the input text, ensuring optimal training outcomes..To ensure the accuracy and relevance of predictions, the product will continuously learn and adapt to user feedback. It may incorporate user interactions and preferences into the training process, enabling personalized predictions that align with individual writing styles and preferences.

The next word prediction system will be designed to handle high volumes of text data and provide real-time suggestions to users. It will be scalable and capable of handling increased computational demands as the user base and the amount of data grow.The product will be developed with a focus on performance, efficiency, and reliability. It will undergo rigorous testing and quality assurance processes to ensure that it meets the highest standards of accuracy and usability.

The product will be designed to support multiple languages, allowing users from different linguistic backgrounds to benefit from accurate next word predictions in their respective languages.The product will strive to be inclusive and accessible to users with diverse needs. It will consider accessibility standards, such as supporting screen readers, keyboard navigation, and adjustable font sizes, to ensure that individuals with disabilities can benefit from the next word prediction functionality.

Overall, the product's perspective is to empower users with an intelligent and efficient text composition tool that enhances their productivity and writing experience. By leveraging the capabilities of neural networks and advanced NLP techniques, the product aims to deliver accurate, context-aware, and intuitive next word predictions.

Product Functions

Our program is the main functional requirement where we will be using a programming language, example C, C++, etc.

- The system shall constantly access the dictionary file. User will give input in the form ofa word (partial or full). Words matching the letters typed so far will be displayed onthe screen.
- If there are no matching words, then system will ask if user wants to add the word todictionary
- After every word prediction cycle completes, the system asks if the user wants to add more words. If yes, another cycle starts, else the system halts.

User Classes and Characteristics

- All types of users-Word Predictor is intended to speed up typing for people from all types of Backgrounds and experience. Moreover, using it requires very little skill set andcan hence, be used by all types of users.

- Open source software developers and contributors-People who have in depth knowledge of programming language used in the project, in order to understand and be able to meaningfully contribute to the source code.

Operating Environment

Since the Word Predictor will be made primarily using cpp, it can be successfully run on all operating systems with some minor changes in the libraries which might be OperatingSystem dependent.

Design and Implementation Constraints

- Code and commands for implementation of above.
- Usage of slang words
- Usage of languages other than English.

Assumptions and Dependencies

- In this we assume that the user does not search the word that does not exist, if the user does then that word will be added in the corpus and affect the searching for other users.
- In this we assume the API (for compiling the c++ code) documentation does notchange in the future.

1. External Interface Requirements

User Interfaces

Users will be using it using any electronic devices such as laptops, mobiles, computers etc. Input device of system will be used to give input to the software such as Code blocks or any online c++ compiler and when the user presses enter, the software shows words, that matches letters entered so far, on the screen of device being used.

Hardware Interfaces

The hardware interface (keyboard, keypad) will be used to enter letters and then after compilation of the code, the list of words will be fetched from the corpus file.

Software Interfaces

Our program uses an external custom corpus which will be accessed throughout the program. In the file, words are stored in the title case without any space between two words. Every capital letter denotes the beginning of a new word and next one denotes end.

Communications Interfaces

This project supports all types of web browsers. We are using a simple yet interactive word prediction system.

2. System Features

System Feature 1 :

System Feature	**Word Prediction Program**
Priority	High
Description	It will suggest relevant words to the user to minimize time and keystrokes.

Functional Requirements	**TRIE data structure implemented in C++ language and a external custom corpus**
Action	This module is activated after the user provides a few letters into the interface.
Result	The system shows the results of word prediction query and the user can choose one of the displayed words or add a word to the corpus.

System Feature 2:

System Feature	**User interaction and feedback**
Priority	Medium
Description	The user can use the program in a more convenient and interactive way
Action	One the user enters a few letters in the portal,the website will display the list of words fromcorpus and other relevant options.
Result	If a new word is searched or added, it will be included into the original dictionary for even better performance in future.
Functional Requirements	A website

3. Other Nonfunctional Requirements:

a. Performance Requirements

- The system should be interactive to users
- The interface is simple and easy to use.
- System is user friendly, self-explanatory and also it is provided with a help guide.
- This system can be used by everyone.
- Speed: The system should be made as fast as possible to reduce response time.
- Throughput: The throughput should be as high as possible. We should be able to attain maximum output in minimum time.
- Resource Utilization: Resources are modified according to user requirements

  .

b. Safety Requirements

- Operation of weekly backups for the database should take place.
- The system should be tough and not prone to breakdowns.

c. Security Requirements

- The administrators maintain the system as per the maintenance contract.
- The system has to be secure from attacks.
- In case of breakdown should be stabilized soon.

d. Quality Attributes

a. Reliability

The solution should provide reliability to the user that the product will run with all the features mentioned in this document are available and executing perfectly. It should be tested and debugged completely. All exceptions should be well handled.

- Try to attain maximum reliability.
- Reliability will also be higher since we try to attain maximum accuracy.
- Maintain proper and updated dictionary files to improve reliability.

b. Accuracy

The product should be able to reach the desired level of accuracy. This prototype version is primarily for proving the concept of the project.

- The information provided in the dictionary files and by the user should be correct.
- Minimize the errors
- All operations will be done correctly to increase the level of accuracy

e. **Business Rules:**

Module Name	**To determine if TRIE is a suitable data structure for word prediction program**

Research Objective	Compare the performance of some other data structures andthen TRIE when used to predict word sand choose the data structure which produces the best result.
Description	After comparing a few studies on prediction using data structures, it was found that TRIE data structure was muchmore efficient than other data structures because of its unique advantages such as hashing, self-balancing, no need for collision handling and ease of printing all words in alphabetical order.
Expected Outcome	Confirmation that TRIE is an efficient data structure for word prediction.

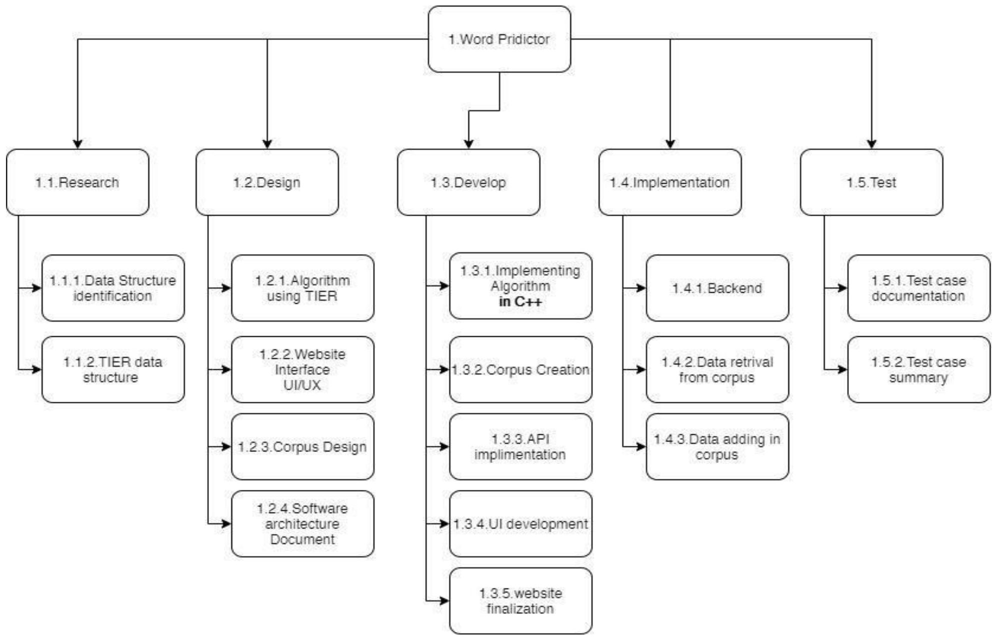
Appendix A: Glossary

Trie: Trie is an efficient information retrieval data structure. Using Trie, search complexities can be brought to optimal limits (key length). If we store keys in binary search tree, a well-balanced BST will need time proportional to M * log N, where M is maximum string length and N is number of keys in the tree. Using Trie, we can search the key in O(M) time.

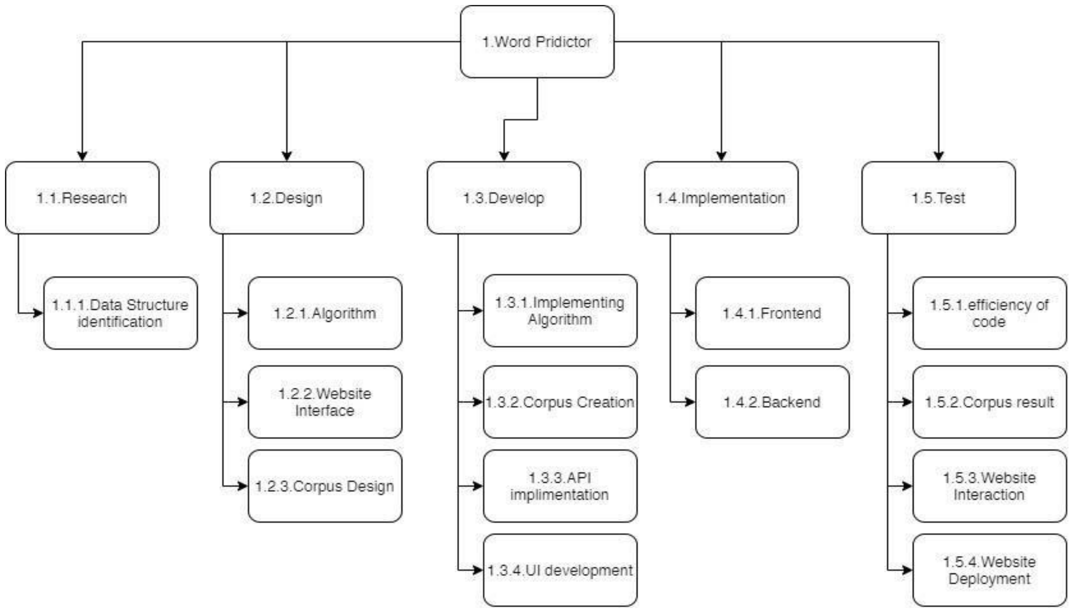
Corpus: It is a collection of written texts, especially a body of writing on a particular subject. Here, it refers to the dictionary of words from which words will be fetched and made available to the user.

2.2.5. Work breakdown structure

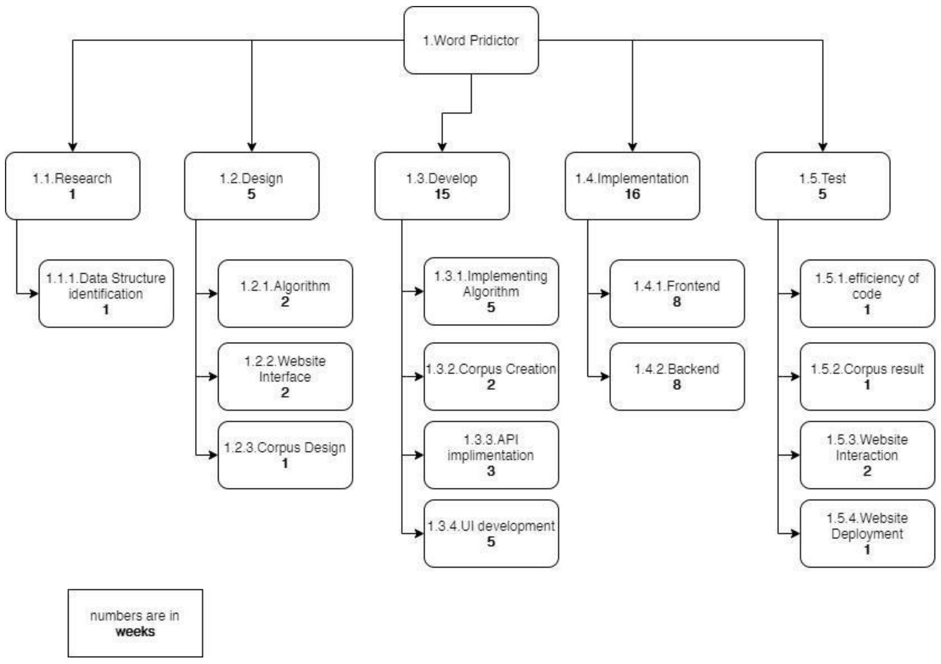
1. Noun-oriented WBS: It is a deliverable-oriented WBS that defines the project work in terms of the components (physical or functional) that make up the deliverable


1.Word Pridictor
1.1.Research
1.1.1.Data Structure identification
1.1.2.TIER data structure
1.2.Design
1.2.1.Algorithm using TIER
1.2.2.Website Interface UI/UX
1.2.3.Corpus Design
1.2.4.Software architecture Document
1.3.Develop
1.3.1.Implementing Algorithm in C++
1.3.2.Corpus Creation
1.3.3.API implimentation
1.3.4.UI development
1.3.5.website finalization
1.4.Implementation
1.4.1.Backend
1.4.2.Data retrival from corpus
1.4.3.Data adding in corpus
1.5.Test
1.5.1.Test case documentation
1.5.2.Test case summary

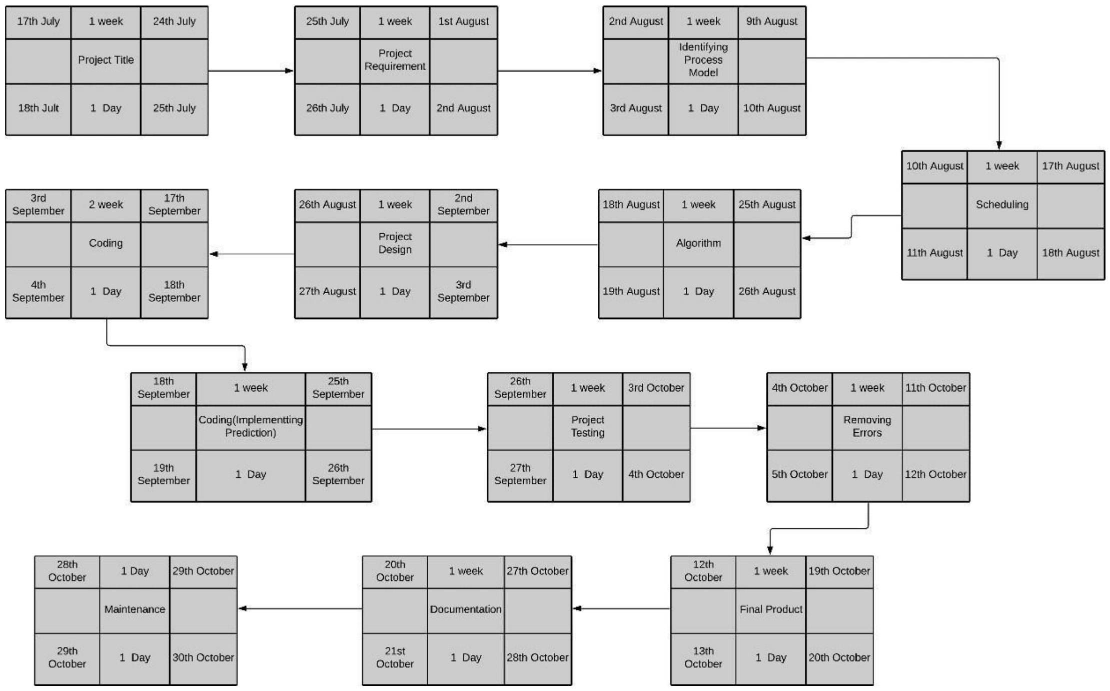
2. Verb-oriented WBS: It is a task-oriented WBS that defines the deliverable of project work in terms of the actions that must be done to produce the deliverable.



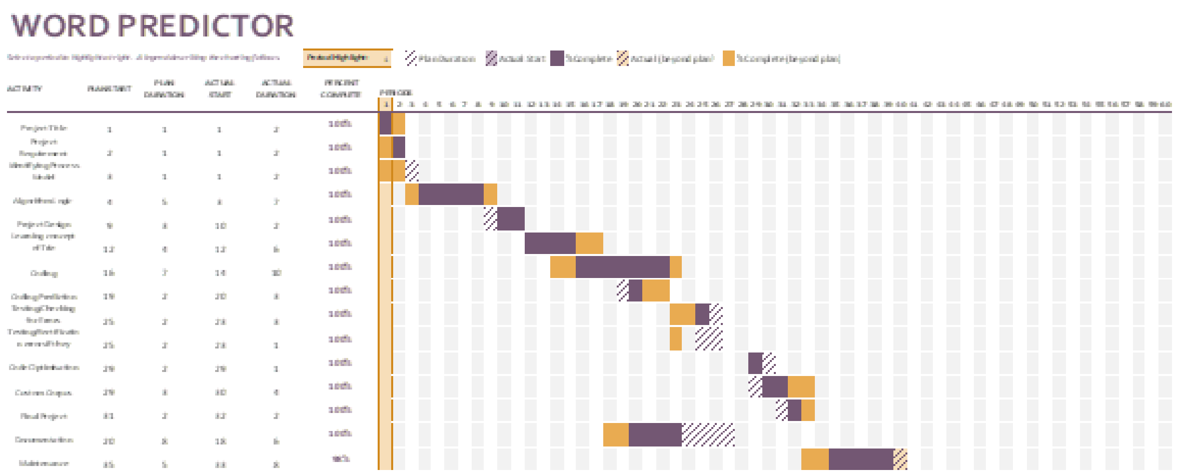
3. Time-Phased WBS: It is a time phased WBS and is typically used for very longprojects. The project is broken down into major phases instead of tasks.



2.2.6. Pert chart



2.2.7. Gantt Chart


WORD PREDICTOR

[illegible]

Period Highlight: 1

▨ Plan Duration ▨ Actual Start ■ % Complete ▨ Actual (beyond plan) ■ % Complete (beyond plan)

ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Project Title	1	1	1	2	100%
Project Requirement	2	1	1	2	100%
Identifying Process Model	3	1	1	2	100%
Algorithm Logic	4	5	3	7	100%
Project Design	9	3	10	2	100%
Learning concept of [illegible]	12	4	12	6	100%
Coding	16	7	14	10	100%
Coding Prediction	19	2	20	3	100%
Testing/Checking for Errors	25	2	23	3	100%
Testing/[illegible]	25	2	23	1	100%
Code Optimization	29	2	29	1	100%
Custom Corpus	29	3	30	4	100%
Final Project	31	2	32	2	100%
Documentation	20	8	18	6	100%
Maintenance	35	5	38	8	90%

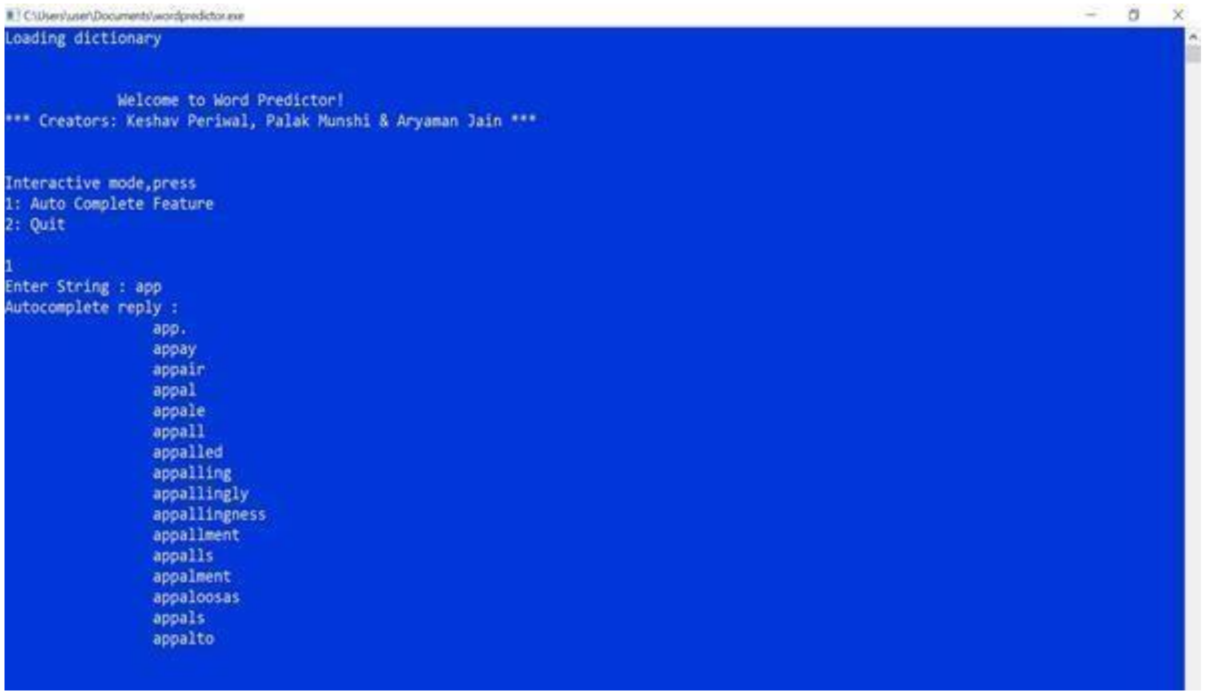
4. Design of the Proposed System

4.1. Introduction

Users will be using it using any electronic devices such as laptops, mobiles,computers etc. Input device of system will be used to give input to the software such as Codeblocks or any online cpp compiler and when the user presses enter, the software shows words, that matches letters entered so far, on the screen of device being used.

4.2. High level Design

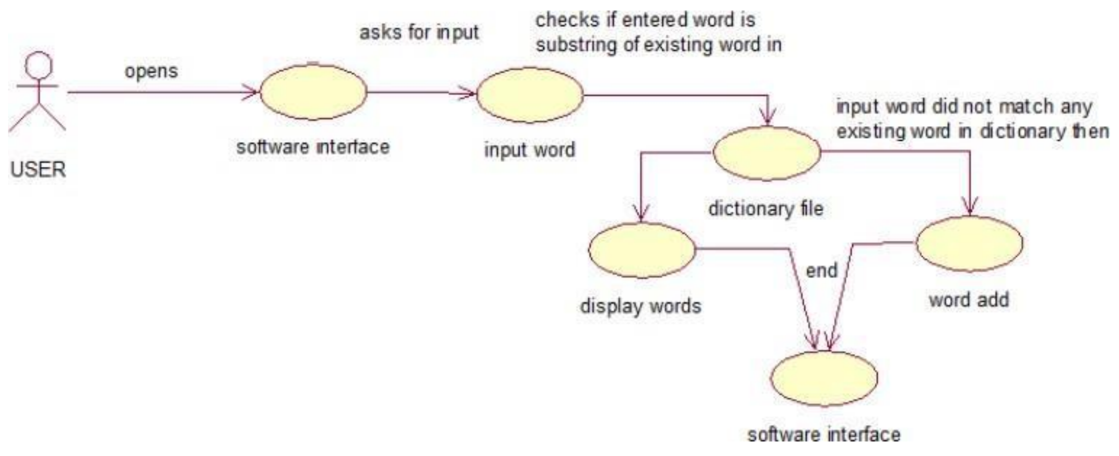
4.2.1. UI design



```
C:\Users\user\Documents\wordpredictor.exe
Loading dictionary


             Welcome to Word Predictor!
*** Creators: Keshav Periwal, Palak Munshi & Aryaman Jain ***


Interactive mode,press
1: Auto Complete Feature
2: Quit

1
Enter String : app
Autocomplete reply :
                 app.
                 appay
                 appair
                 appal
                 appale
                 appall
                 appalled
                 appalling
                 appallingly
                 appallingness
                 appallment
                 appalls
                 appalment
                 appaloosas
                 appals
                 appalto
```

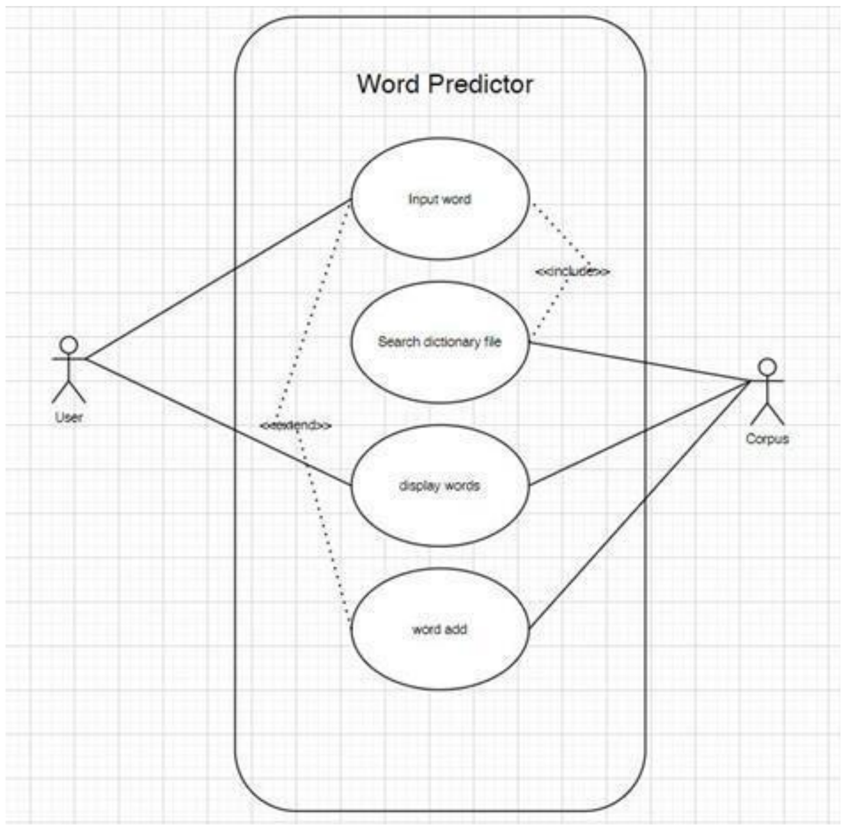
.4.3. Detailed Design (ER Diagram/UML Diagram/Mathematical Modeling)

4.3.1. ER Diagram

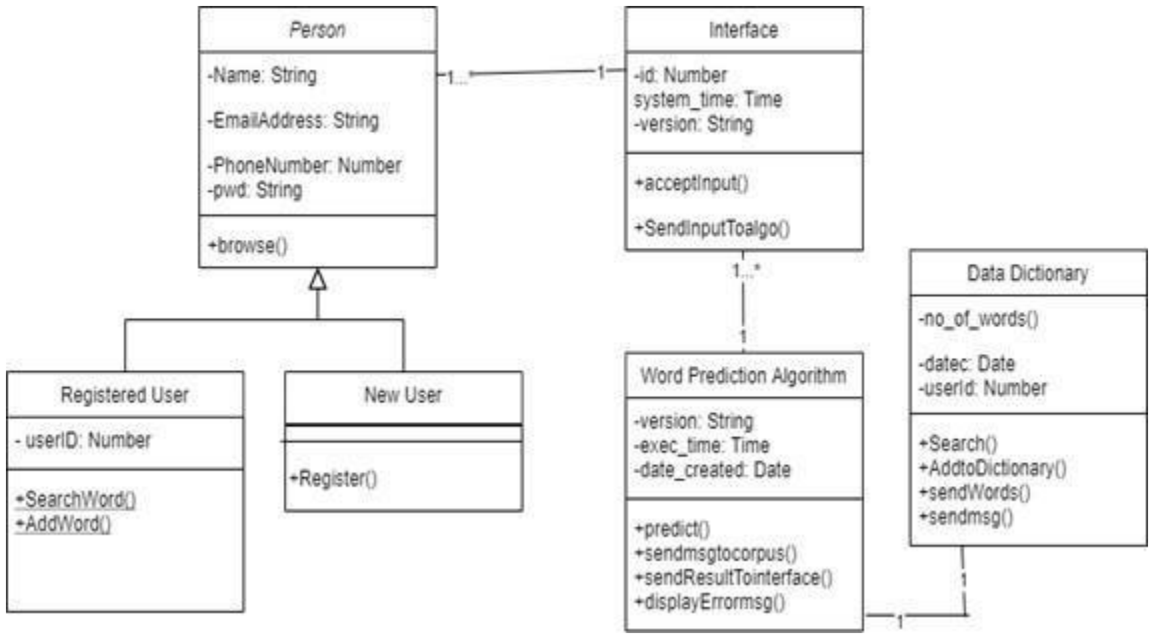

USER
opens
software interface
asks for input
input word
checks if entered word is substring of existing word in
dictionary file
input word did not match any existing word in dictionary then
display words
end
word add
software interface

4.3.2. UML diagram (Use case, class, State chart, Activity and interactiondiagrams)

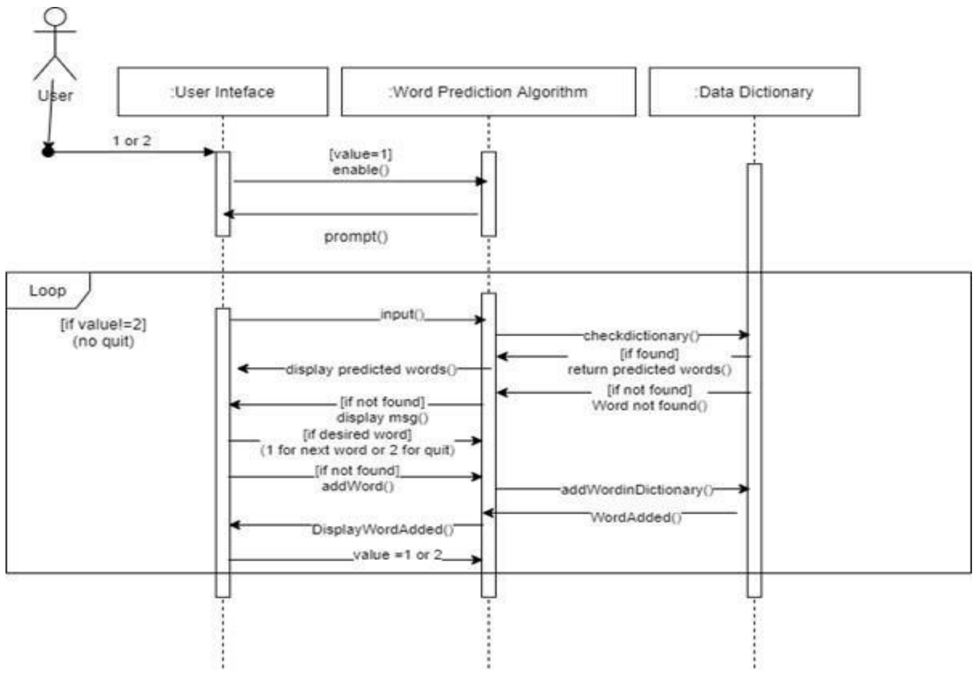
A) Use Case Diagram



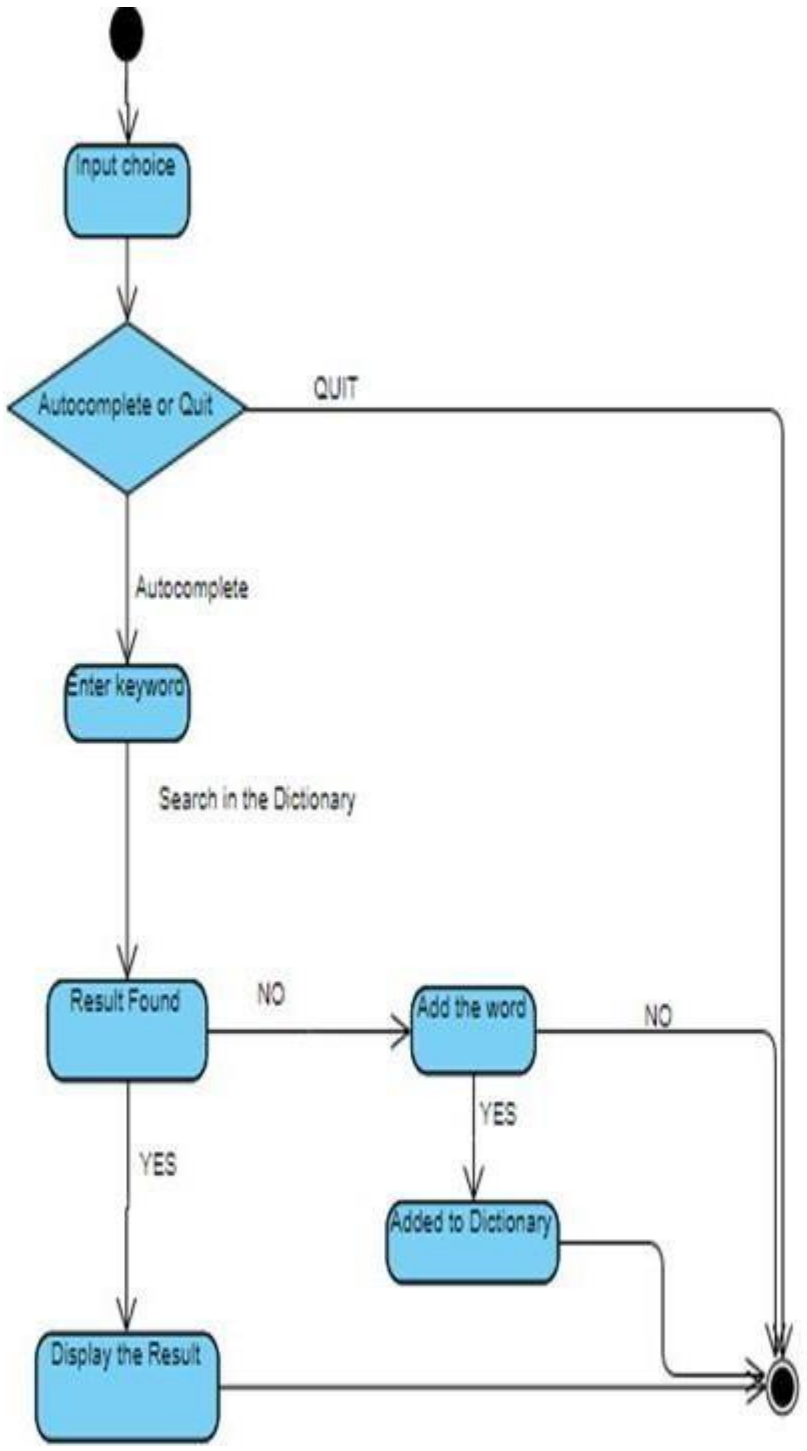
A) Class Diagram



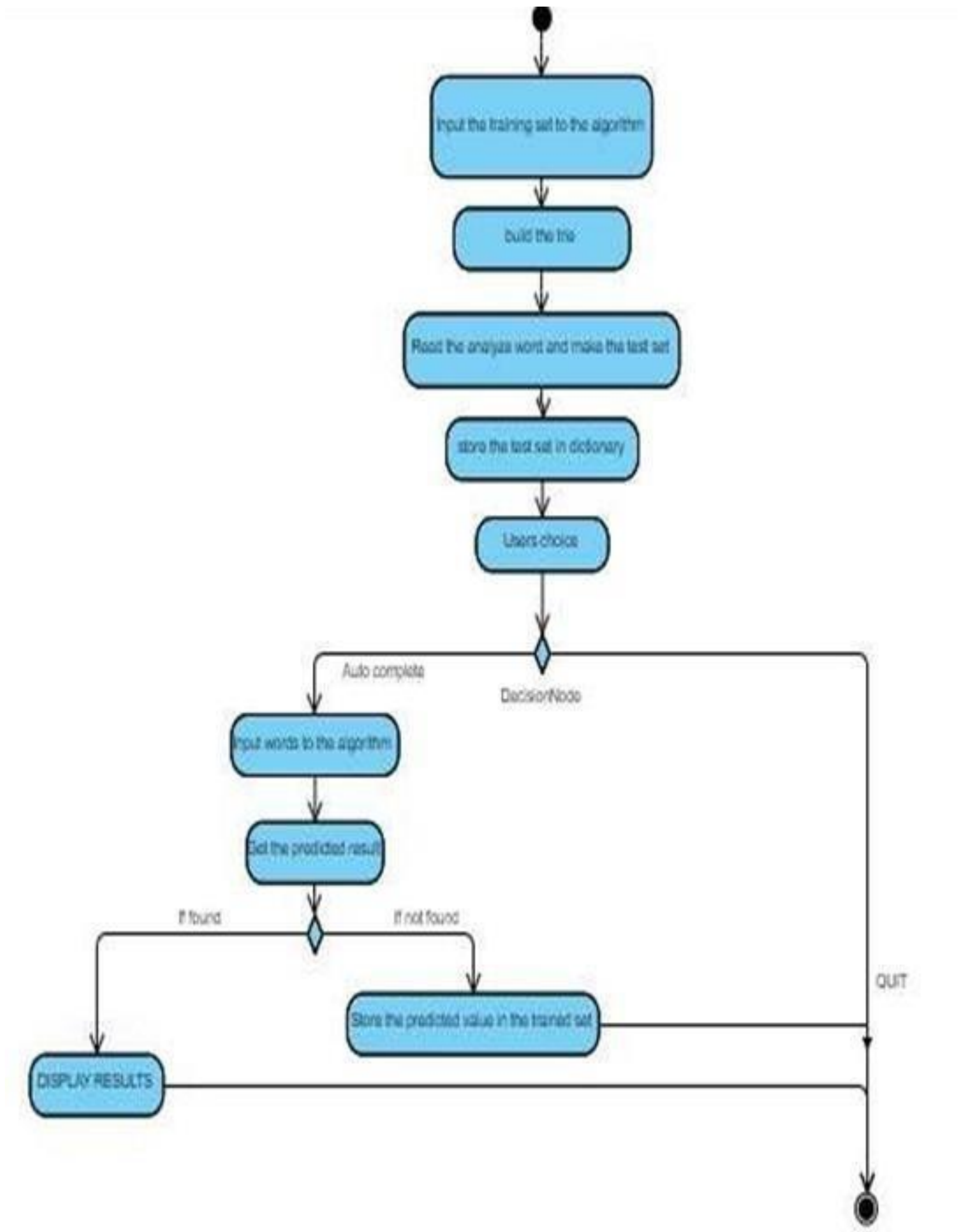
B) Sequence Diagram



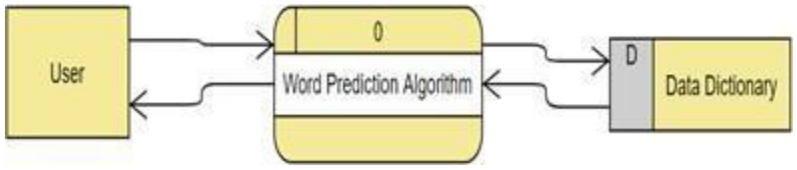
C) State Transition Diagram



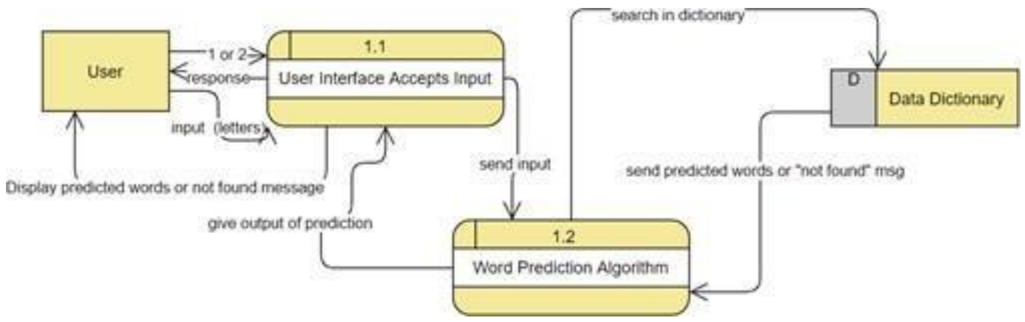
D) Activity Diagram


Input the training set to the algorithm
build the trie
Read the analyze word and make the test set
store the test set in dictionary
Users choice
Auto complete
DecisionNode
Input words to the algorithm
Get the predicted result
If found
If not found
QUIT
Store the predicted value in the trained set
DISPLAY RESULTS

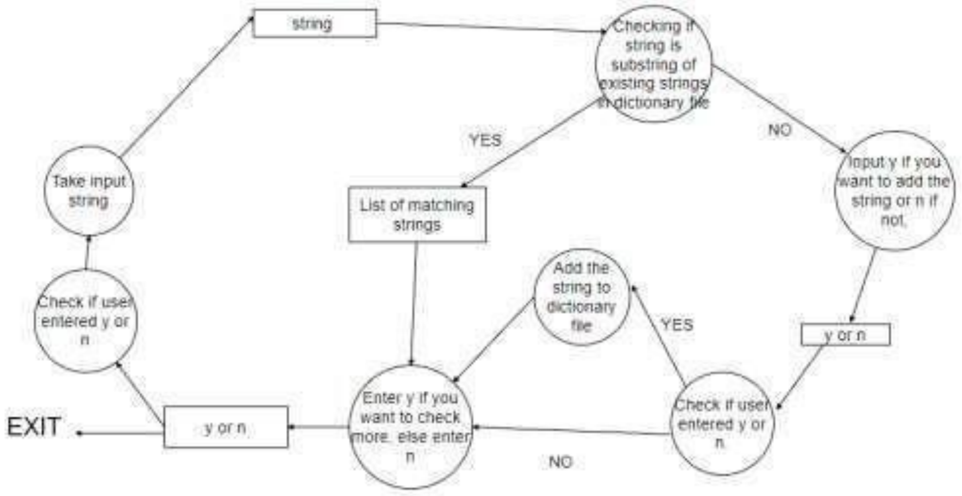
E) Data Flow Diagram Level 0



F) Data Flow Diagram Level 1


User
1 or 2
response
1.1
User Interface Accepts Input
input (letters)
Display predicted words or not found message
give output of prediction
send input
search in dictionary
D
Data Dictionary
send predicted words or "not found" msg
1.2
Word Prediction Algorithm

G). Data Flow Diagram Level 2

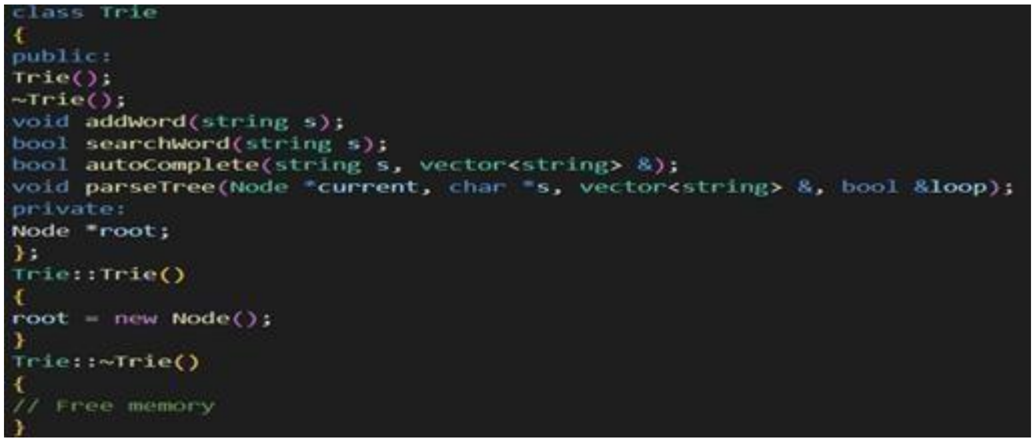

string
Checking if string is substring of existing strings in dictionary file
YES
NO
Take input string
List of matching strings
Input y if you want to add the string or n if not.
Add the string to dictionary file
YES
y or n
Check if user entered y or n
EXIT
y or n
Enter y if you want to check more, else enter n
NO
Check if user entered y or n

5. Implementation and Testing (Snap shots with description)

5.1 Implementation details (snapshots)

Module 1: Class Trie

Code Snippet:

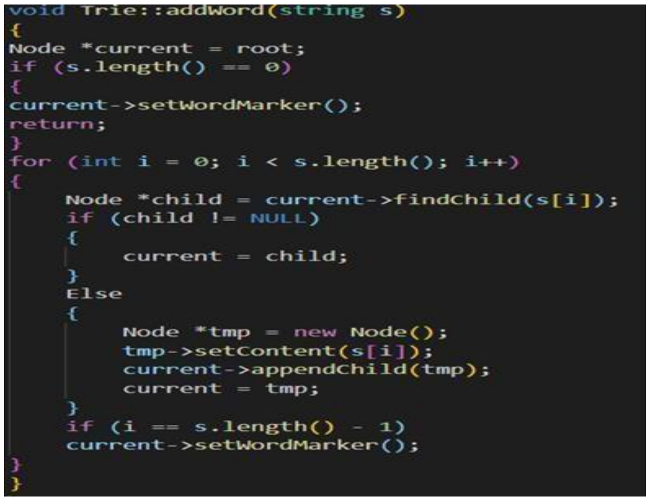


```
class Trie
{
public:
Trie();
~Trie();
void addWord(string s);
bool searchWord(string s);
bool autoComplete(string s, vector<string> &);
void parseTree(Node *current, char *s, vector<string> &, bool &loop);
private:
Node *root;
};
Trie::Trie()
{
root = new Node();
}
Trie::~Trie()
{
// Free memory
}
```

Description: This is the main structure of our code. In this class Triewe have 4 functions addWord, searchWord, autoComplete, parseTree.

Module 2: addWord()

Code Snippet:

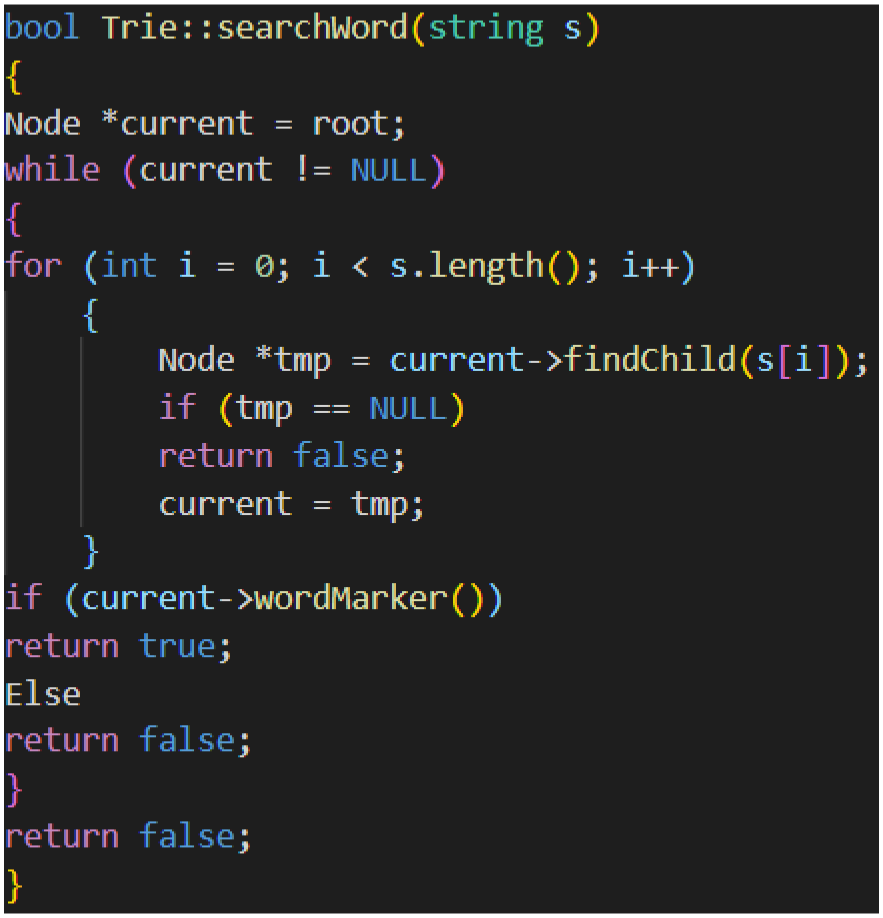


```
void Trie::addWord(string s)
{
Node *current = root;
if (s.length() == 0)
{
current->setWordMarker();
return;
}
for (int i = 0; i < s.length(); i++)
{
    Node *child = current->findChild(s[i]);
    if (child != NULL)
    {
        current = child;
    }
    Else
    {
        Node *tmp = new Node();
        tmp->setContent(s[i]);
        current->appendChild(tmp);
        current = tmp;
    }
    if (i == s.length() - 1)
    current->setWordMarker();
}
}
```

Description: This function helps us to add the word in the corpus if the entered word from the user is not available in the corpus. In for loop the we are findingwhether the word is there or not, if there then it will return current as a word and if not then else statement will append the word inthe corpus.

Module 3: Function searchWord

Code Snippet:

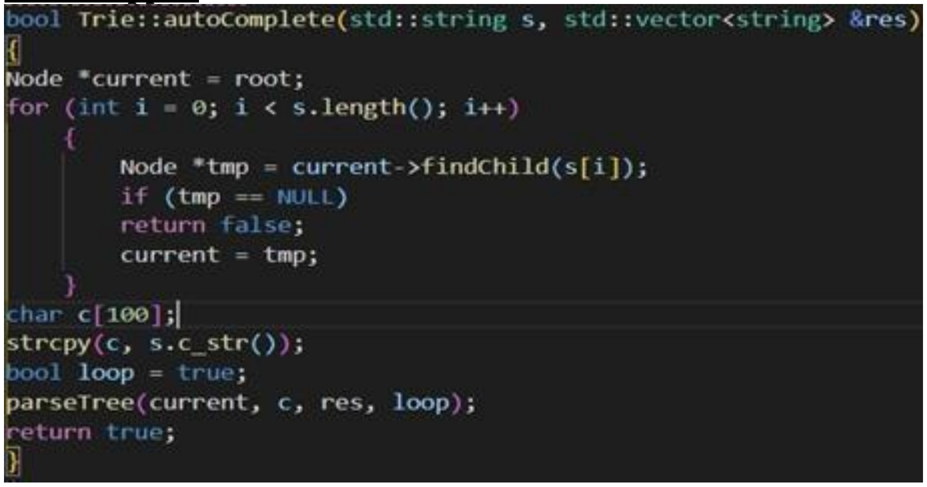


```
bool Trie::searchWord(string s)
{
Node *current = root;
while (current != NULL)
{
for (int i = 0; i < s.length(); i++)
    {
        Node *tmp = current->findChild(s[i]);
        if (tmp == NULL)
        return false;
        current = tmp;
    }
if (current->wordMarker())
return true;
Else
return false;
}
return false;
}
```

Description: This function is used to search the word entered by the user. In for statement find Child is used to find the word in the corpus, if the word is found then it will return it as current or else it will return false.

Module 4: Function autoComplete

Code Snippet:

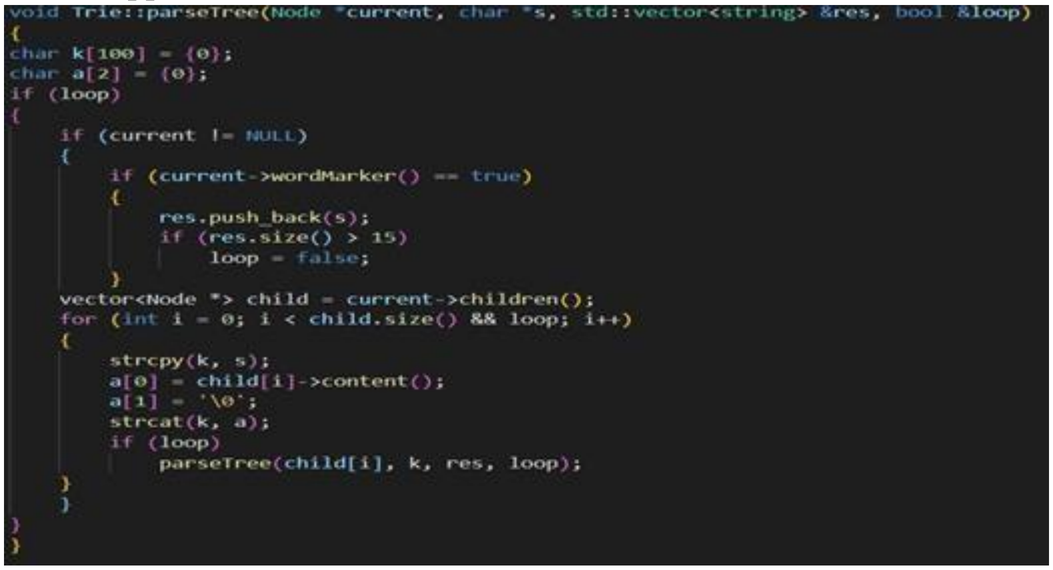


```cpp
bool Trie::autoComplete(std::string s, std::vector<string> &res)
{
Node *current = root;
for (int i = 0; i < s.length(); i++)
    {
        Node *tmp = current->findChild(s[i]);
        if (tmp == NULL)
        return false;
        current = tmp;
    }
char c[100];
strcpy(c, s.c_str());
bool loop = true;
parseTree(current, c, res, loop);
return true;
}
```

Description: This function is used to concatenate the words and form thewords. In for statement first it will try to find the words with the string entered by the user and after that it concatenates the words and shows theoutput.

Module 5: Function parseTree

Code Snippet:

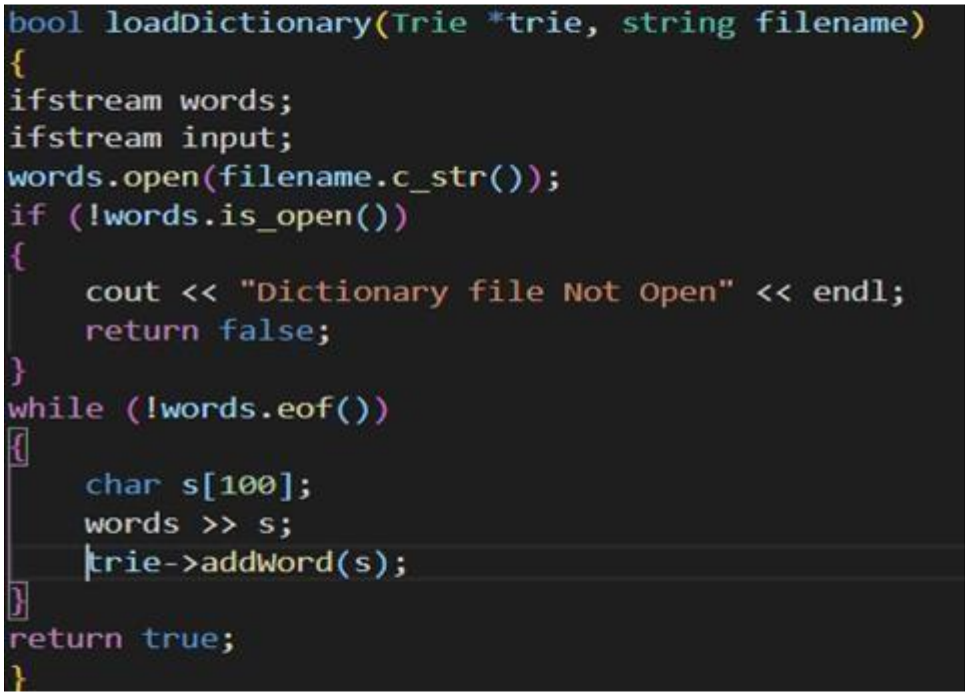


```cpp
void Trie::parseTree(Node *current, char *s, std::vector<string> &res, bool &loop)
{
char k[100] = {0};
char a[2] = {0};
if (loop)
{
    if (current != NULL)
    {
        if (current->wordMarker() == true)
        {
            res.push_back(s);
            if (res.size() > 15)
                loop = false;
        }
    vector<Node *> child = current->children();
    for (int i = 0; i < child.size() && loop; i++)
    {
        strcpy(k, s);
        a[0] = child[i]->content();
        a[1] = '\0';
        strcat(k, a);
        if (loop)
            parseTree(child[i], k, res, loop);
    }
    }
}
}
```

Description: This function is used to parse the tree and concatenate the input string with different letters and try to form a word and then show them as anoutput.

Module 6: Function loadDictonary

Code Snippet:



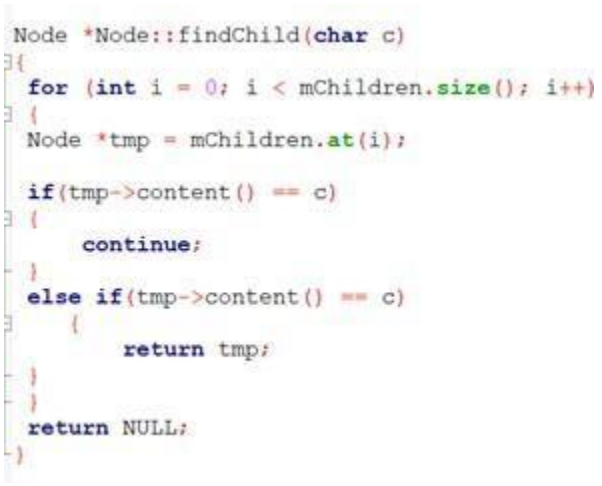
```
bool loadDictionary(Trie *trie, string filename)
{
ifstream words;
ifstream input;
words.open(filename.c_str());
if (!words.is_open())
{
    cout << "Dictionary file Not Open" << endl;
    return false;
}
while (!words.eof())
{
    char s[100];
    words >> s;
    trie->addWord(s);
}
return true;
}
```

Description: This function loads the corpus and then tries torun all the functions explained above.

Code Optimization

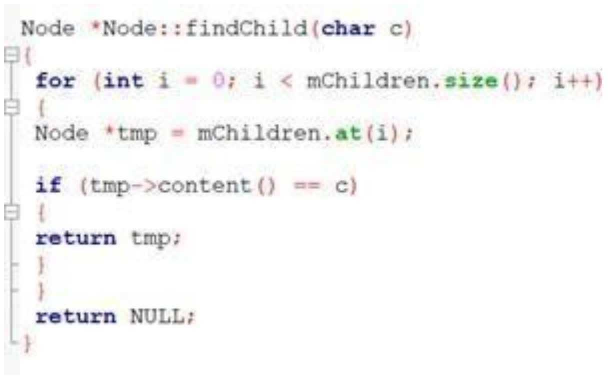
Various Code optimization techniques are- Constant Folding, Constant Propagation, Common Subexpression elimination, Code movement, Dead code elimination, Strength reduction.The ones demonstrated below incorporate Dead Code Elimination and Constant Propagation.

1) Before Optimsation:



```
Node *Node::findChild(char c)
{
 for (int i = 0; i < mChildren.size(); i++)
 {
 Node *tmp = mChildren.at(i);

 if(tmp->content() == c)
 {
     continue;
 }
 else if(tmp->content() == c)
    {
        return tmp;
 }
 }
 return NULL;
}
```

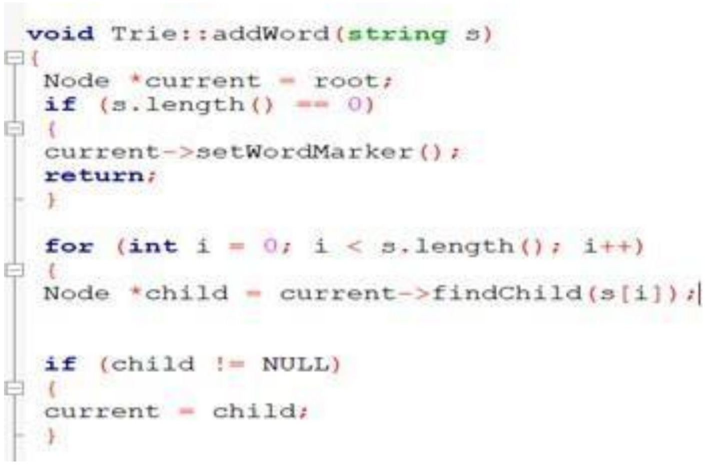
After Optimization:



```
Node *Node::findChild(char c)
{
 for (int i = 0; i < mChildren.size(); i++)
 {
 Node *tmp = mChildren.at(i);

 if (tmp->content() == c)
 {
 return tmp;
 }
 }
 return NULL;
}
```

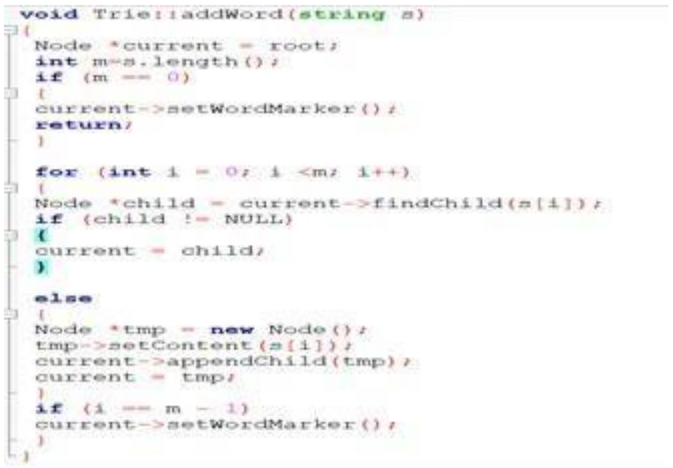
The optimization: The if-else conditions are optimized to just an if condition without any loss in logic and thus time and spacecomplexities are reduced.

2) Before Optimization:



```
void Trie::addWord(string s)
{
 Node *current = root;
 if (s.length() == 0)
 {
 current->setWordMarker();
 return;
 }

 for (int i = 0; i < s.length(); i++)
 {
 Node *child = current->findChild(s[i]);


 if (child != NULL)
 {
 current = child;
 }
```

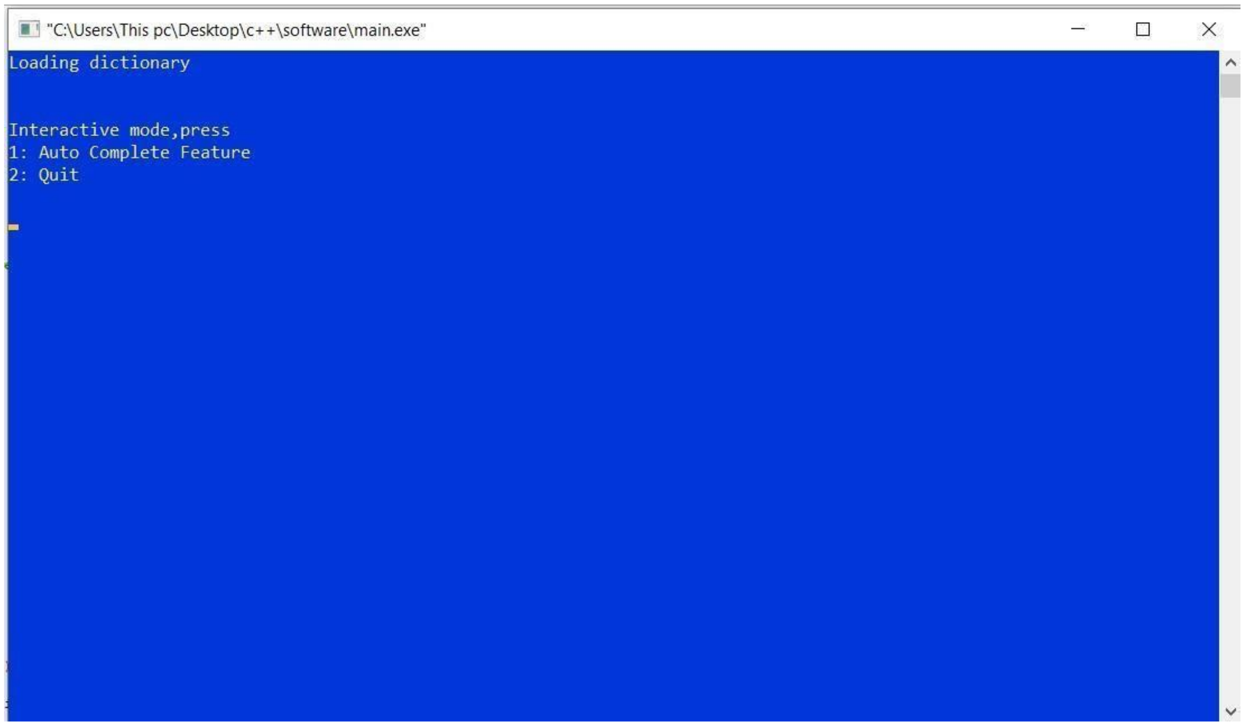
After Optimization:



```
void Trie::addWord(string s)
{
Node *current = root;
int m=s.length();
if (m == 0)
{
current->setWordMarker();
return;
}

for (int i = 0; i <m; i++)
{
Node *child = current->findChild(s[i]);
if (child != NULL)
{
current = child;
}

else
{
Node *tmp = new Node();
tmp->setContent(s[i]);
current->appendChild(tmp);
current = tmp;
}
if (i == m - 1)
current->setWordMarker();
}
}
```

The optimization: The value of s.length() is stored in a variable m and the need to calculate s.length again in thefuture is eliminated because every time the variable ‘m’ isreferred instead.

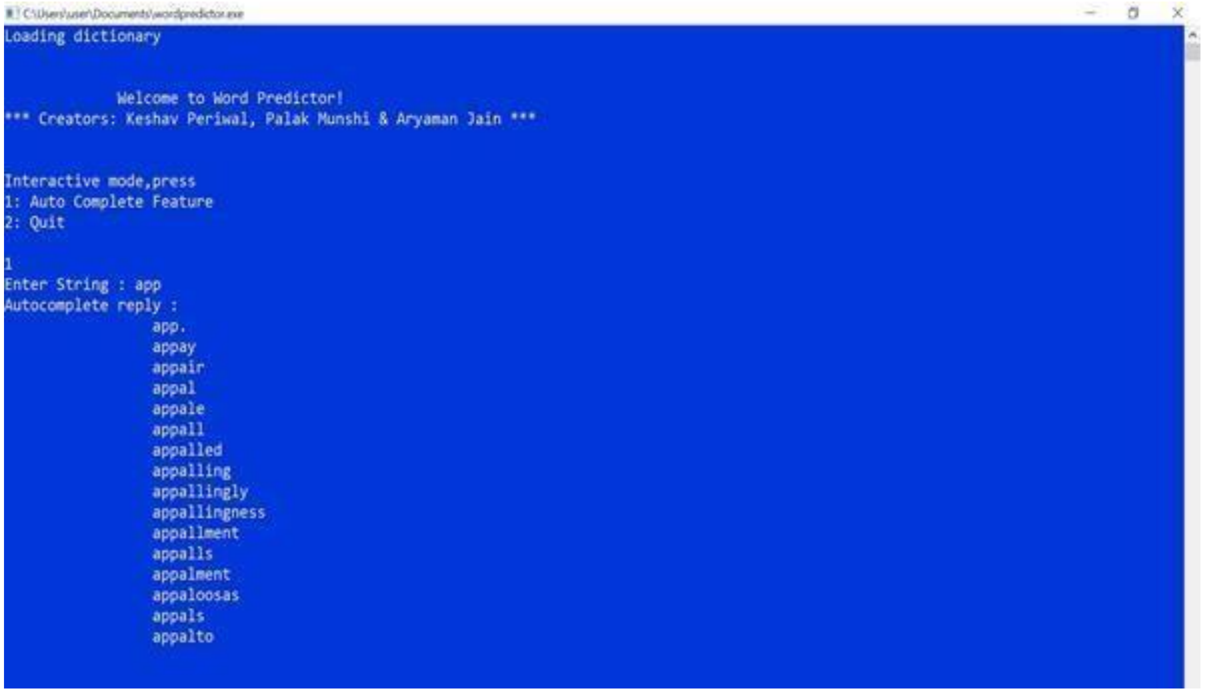
OUTPUT:



"C:\Users\This pc\Desktop\c++\software\main.exe"

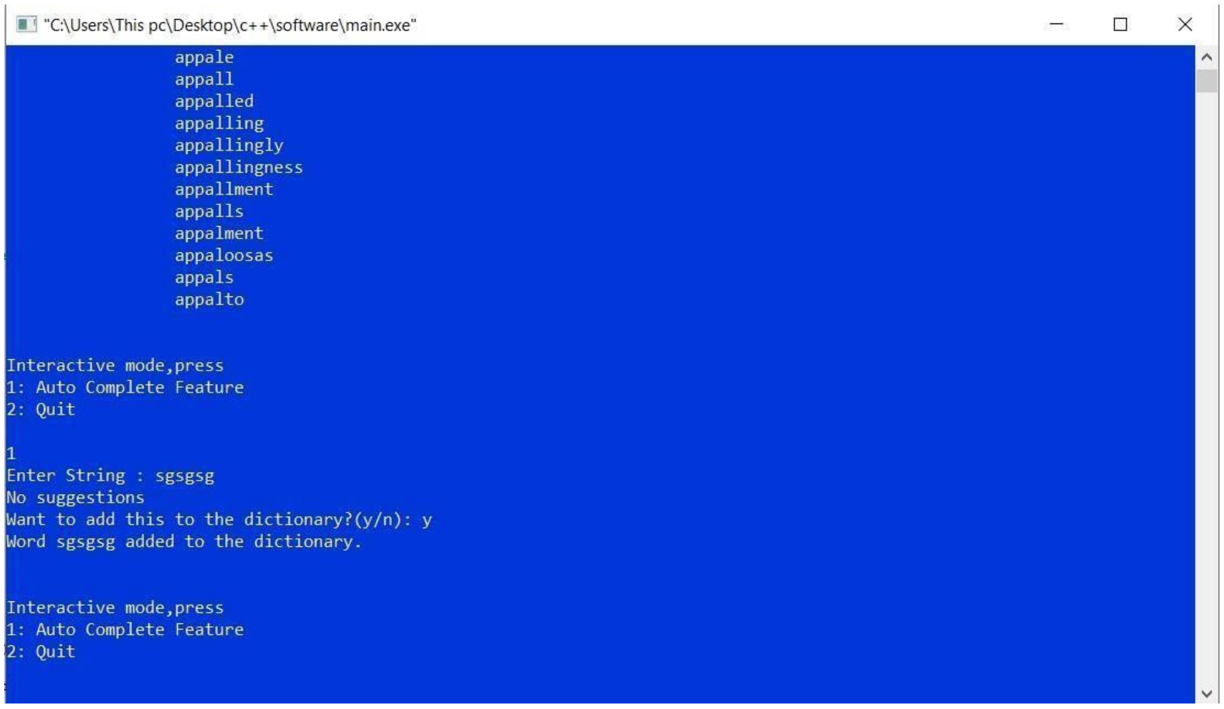
```
Loading dictionary


Interactive mode,press
1: Auto Complete Feature
2: Quit
```

First the program asks if user wants to go for ‘1. Autocomplete’ or wants to ‘2. Quit’ the program.If user chooses ‘1’, he’ll get this output:

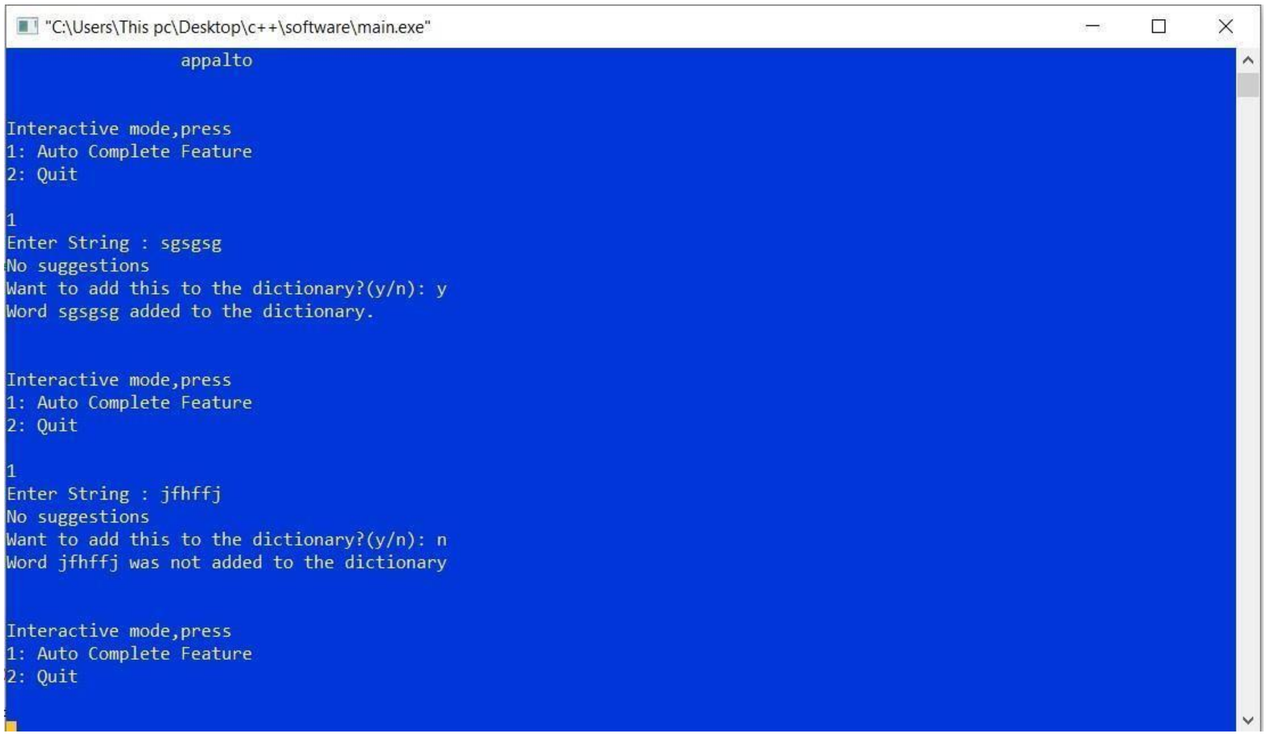


```
C:\Users\user\Documents\wordpredictor.exe
Loading dictionary


            Welcome to Word Predictor!
*** Creators: Keshav Periwal, Palak Munshi & Aryaman Jain ***


Interactive mode,press
1: Auto Complete Feature
2: Quit

1
Enter String : app
Autocomplete reply :
                app.
                appay
                appair
                appal
                appale
                appall
                appalled
                appalling
                appallingly
                appallingness
                appallment
                appalls
                appalment
                appaloosas
                appals
                appalto
```

The program asks for the string. And displays matching words. If the input string does not have any matches in the dictionary file, it shows this:



```
"C:\Users\This pc\Desktop\c++\software\main.exe"
                 appale
                 appall
                 appalled
                 appalling
                 appallingly
                 appallingness
                 appallment
                 appalls
                 appalment
                 appaloosas
                 appals
                 appalto


Interactive mode,press
1: Auto Complete Feature
2: Quit

1
Enter String : sgsgsg
No suggestions
Want to add this to the dictionary?(y/n): y
Word sgsgsg added to the dictionary.


Interactive mode,press
1: Auto Complete Feature
2: Quit
```



```
                 appalto


Interactive mode,press
1: Auto Complete Feature
2: Quit

1
Enter String : sgsgsg
No suggestions
Want to add this to the dictionary?(y/n): y
Word sgsgsg added to the dictionary.


Interactive mode,press
1: Auto Complete Feature
2: Quit

1
Enter String : jfhffj
No suggestions
Want to add this to the dictionary?(y/n): n
Word jfhffj was not added to the dictionary


Interactive mode,press
1: Auto Complete Feature
2: Quit
```

The user is asked if they want to add the string into the dictionary file. If users will press ‘y’, it asks for a full word that’s to be added. If the user presses ‘n’, the program again asks if theywant to ‘1. Autocomplete’ or ‘2. Quit’.

5.1. Testing

5.1.1. Types of Testing

- For our algorithm, we performed the blackbox and the whitebox testing. Blackbox Testing is defined as is a software testing method in which the functionalities of software applications are tested without having knowledge of internal code structure,implementation details and internal paths.

- Black Box Testing mainly focuses on input and output of software applications and itis entirely based on software requirements and specifications. The test cases are summarized later.

- White Box Testing is a software testing technique in which internal structure, designand coding of software are tested to verify flow of input-output and to improve design,usability and security.

5.1.2. . Test cases

Testing:

Test Case	Expected Output	Actual Output	PASS/FAIL
1.On selecting auto complete feature	It should ask for “Enter String”	“Enter String”, appears	PASS
2.On selecting quit	It should end the program	The program ends	PASS
3.Entering a few letters of the word	Should display the list of words		
3.a.”app”	List of all the wordsstarting with “app”	List of all words	PASS
3.b.”hel”	List of all the wordsstarting with “hel”	List of all words	PASS
4.Enter the word which is not in thecorpus	Should show a massage ”No suggestion”		PASS
4.a.”serp”	List of all the wordsstarting with “serp”	List of all words	PASS
4.b.”sdrdedf”(jibrish)		“No suggestions”	PASS
5.Ask to add the word to the corpus	Adds word tocorpus[y/n]	Add word to corpus[y/n]	PASS
5.a..on typing [y]	The word should be added to the corpus and a message should be presented with “the “word” is added	The “word” is added	PASS
5.b. On typing [n	The word should not be added to the corpus and a message should be presented with “the “word” is not added	The “word” is notadded	PASS

6. Conclusion:

Word Predictor has applications in messaging applications like whatsapp, web search engines, word processors, command like interpreters etc. The original purpose of word prediction software was to help people with physical disabilities increase their typing speed,[1] as well as to help them decrease the number of keystrokes needed in order to complete a word or a sentence. Thus, in this front we developed our own program for word prediction using data structure trie which definitely increases efficiency of the user by at least 10%.

Limitations and Scope for future Work:

Advantages of our Algorithm:

- A word predictor has applications in various areas like texting, search engines and thus,can help address this issue for a wide demographic.
- Existing programs can be made more efficient using data structures like TRIE.
- Using data structure Trie, word prediction can be made more efficient by at least10%.
- With data structure Trie, the time complexity for retrieval is reduced and it is much easier to print words in alphabetical order. Both of these are useful for building an efficient and fast word-prediction algorithm.

Limitations of our Algorithm:

- Instead of displaying suggestions when prompted, the algorithm could be tried to make dynamic which displays new suggestions as each new letter is typed.
- The corpus can be made much more extensiv

Future Scope:

The potential for next word prediction using neural networks is bright, with many opportunities for growth and advancement. Here are some crucial points to think about:

1. Enhanced Context Modelling: Current next word prediction algorithms frequently base their forecasts on local context, such as the last few words. To increase the precision and applicability of predictions, future research can concentrate on including wider context windows or even document-level information.

2. Deep Learning designs: Researching more sophisticated deep learning designs, like transformer models or graph neural networks, may improve the ability to anticipate the words that will come after a given word. With the use of these designs, complicated and distant dependencies that are present in the text can be captured, producing more precise predictions.

3. Multimodal Approaches: Next word prediction algorithms can be improved by integrating multimodal input, such as text mixed with visuals or audio. Models can better comprehend the context and produce predictions that are more relevant to it by utilising several modalities.

4. User-specific Adaptation: One of the most intriguing areas of research is the creation of individualised next word prediction models that take into account unique users' writing styles, preferences, and subject interests. Next word prediction systems can be made far more accurate and user-friendly by adapting them specifically for each user.

5. Domain-specific Prediction: Developing next word prediction models specifically for a given domain, like as legal or medical literature, can produce predictions that are more precise and specialised within that domain. To make suggestions that are more pertinent, these models can be used to add domain-specific grammatical patterns and terminologies.

6. Low-resource Languages: The majority of next word prediction algorithms now in use concentrate on frequently used languages. The development of models for low-resource languages, when there is a lack of training data, can be the focus of future research. This may make next word prediction easier to use and more beneficial for speakers of a wider range of languages.

7. Real-time Prediction: For real-time applications like text editors or chatbots, improving the effectiveness and speed of next word prediction models is essential. Future developments should work to speed up predictions and cut down on inference time without sacrificing accuracy.

7 . References

- Trnka, Keith & McCaw, John & Yarrington, Debra & Mccoy, Kathleen & Pennington, Christopher. (2009). User Interaction with Word Prediction: The Effects of Prediction Quality. TACCESS. 1. 10.1145/1497302.1497307.
- S. P. Singh, A. Kumar, D. C. Mandad and Y. Jadwani, "Word and phrase prediction tool for English and Hindi language," 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), Chennai, 2016, pp. 1485-1488,doi: 10.1109/ICEEOT.2016.7754930.
- Aliprandi, Carlo & Carmignani, Nicola & Deha, Nedjma & Mancarella, Paolo & Rubino, Michele & Srl, Synthema & Pisa, & Italy,. (2008). Advances in NLP appliedto Word Prediction.
- Even-Zohar, Yair & Roth, Dan. (2002). A Classification Approach to Word Prediction.
- Singh, Arjun & Stremmel, Joel. (2020). Pretraining Federated Text Models for NextWord Prediction.
- Al-Mubaid, Hisham. (2007). A Learning-Classification Based Approach for WordPrediction. Int. Arab J. Inf. Technol.. 4. 264-271.
- D. Nagalavi and M. Hanumanthappa, "N-gram Word prediction language models toidentify the sequence of article blocks in English e-newspapers," 2016 InternationalConference on Computation System and Information Technology for Sustainable Solutions (CSITSS), Bangalore, 2016, pp. 307-311, doi: 10.1109/CSITSS.2016.7779376.