

TEXT PREDICTOR USING N-GRAMS

Group Members:

Janhavi Anap (19102043)

Riddhi Narkar (19102003)

Abstract

One of the major technological marvels of this tech-infused 21st century is the ability to connect digitally a popular method of expressing oneself is through text medium.

Often, typing can be prone to errors and not much efficient and quick when it comes to long typing sessions. A text predictor is a piece of software which suggests grammatically correct words after processing your input.

This project implements it by using the concepts like “n-grams” and “markov chain model”.

Problem Definition

One important aspect of effective and fast communication is proper grammar, and choice of words, spellings and more meaningful sentences.

A text predictor not only focuses on suggesting next words which suit your style, but also to the fact that how much the suggested word would make a complete sense in the context of the information being conveyed through the sentence being typed. If a software would be trained to your style of grammar and your diction, and which would adapt the changes of the language you try to adapt in yourself would be an extremely powerful tool and would cater to a wide range of users.

In this project, we developed a text predictor using python using dictionaries, Tkinter, and a python library Gingerit.

Introduction

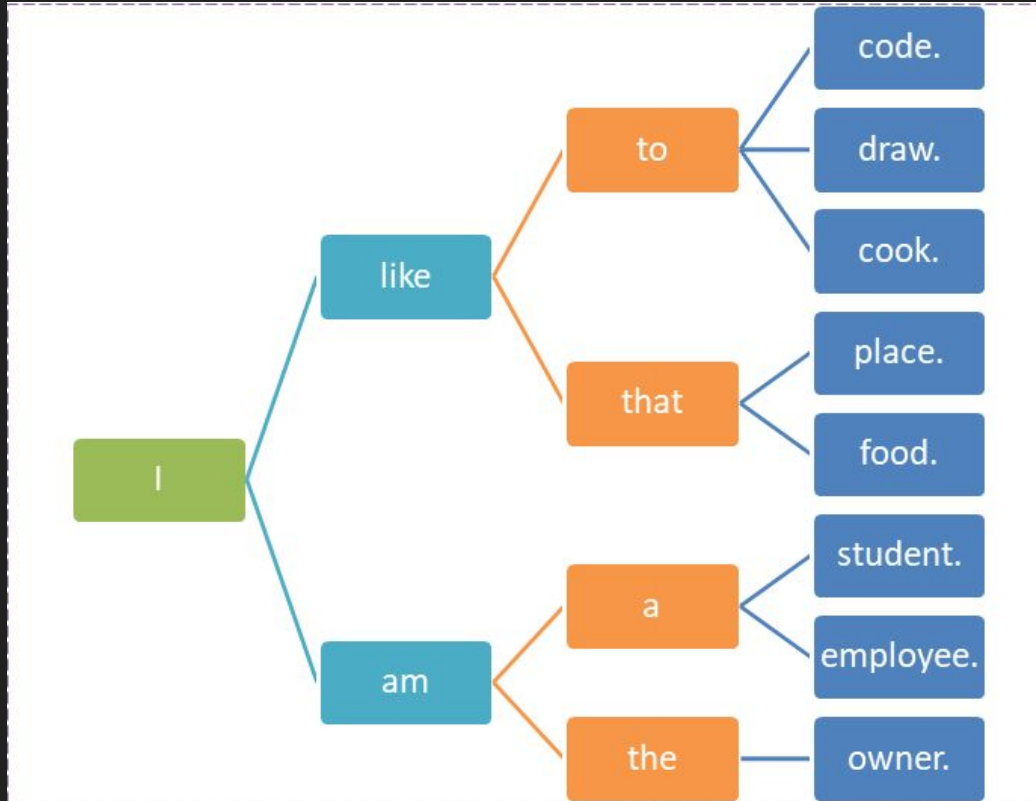
- The word-prediction is implemented using “n-gram” and “markov chain model”. The text input from the user is added in the corpus. Hence the corpus on which the predictor is trained is always up to date.
- The “n-gram” model that we have used has the value of ‘n’ as 3. This is also known as trigrams i.e. the 3rd word is predicted by taking the last two word into account
- For better user experience, tkinter package is used as the GUI

Module Description and Implementation

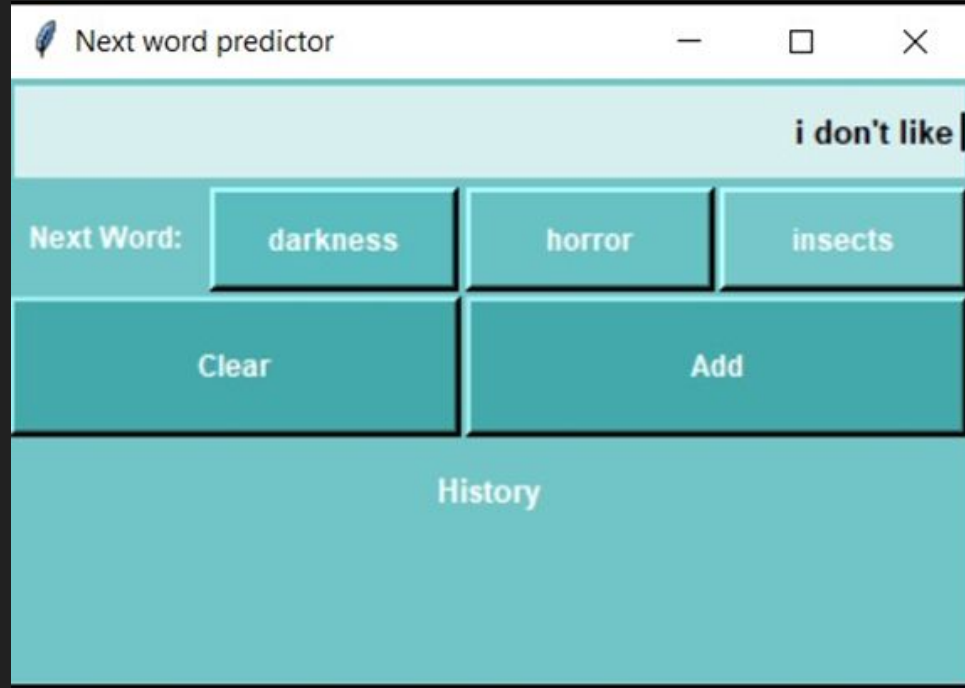
Word Prediction

- The markov model is trained on a previously grammatically correct corpus
- Every time user inputs some any text it is corrected grammatically and added to the corpus using file handling
- The markov model is then trained on this corpus every time before the text-predictor is used keeping the model updated with the user preferences
- The markov model is implemented using nested-dictionary
- The prediction is based on the previous two words and then the top three frequently occurring words are suggested to the user

Markov Chain Model



Real-time prediction



A screenshot of a web application titled "Next word predictor". The interface features a text input field at the top containing the phrase "i don't like". Below this, a section labeled "Next Word:" displays three suggested words: "darkness", "horror", and "insects", each in its own button. Underneath these suggestions are two larger buttons labeled "Clear" and "Add". At the bottom of the interface is a large, empty rectangular area labeled "History".

Next word predictor

i don't like

Next Word:

darkness

horror

insects

Clear

Add

History

Add input text to the history

Next word predictor

i don't like horror films

Next Word: - - -

Clear Add

History

Next word predictor

Next Word: i what hello

Clear Add

History
i don't like horror films

Grammar and spelling Check

Next word predictor

i will studied tomorrow

Next Word:

Clear Add

History

i don't like horror films
i like to read

Next word predictor

i will studied tomorrow

Next Word:

Clear Add

History

i don't like horror films
i like to read

Next word predictor

i am sitting under an treeee

Next Word: - - -

Clear Add

History

i don't like horror films
i like to read
i will study tomorrow

Next word predictor

Next Word: i what hello

Clear Add

History

:
i will study tomorrow
i am sitting under a tree

Future scope

In the project we created, our database was updated with new sentences each time we made a choice based on the predictions, or even if we added a new word of our own choice, thus ignoring the predictions. But the changes reflect in the corpus after we restart the application. Since, we are using file handling, Python doesn't update the corpus text file until the program is terminated. This can be, of course, eliminated by developing a database using MySQL or more efficient technologies. This will make the prediction faster and more robust, as the corpus would immediately be reflected with the changes.

Conclusion

Word prediction, thus, can have a major impact when it comes to using a language in its correct sense.

It can be deployed to use for teaching motor skills to children who are learning a new language, but allowing them to grasp the core semantics intuitively, without having the need to write.

It allows one to use and understand the language, and document anything with a very less error margin.