VATC - Virtual Assistant and Editor for Coding, Debugging, Documenting and Referring

Jawahar Sai Nathani
Sagar Reddy P
Sridhar Chimalakonda
Sriram
cs18b023@iittp.ac.in
cs18b025@iittp.ac.in
ch@iittp.ac.in
cs20s503@iittp.ac.in

ABSTRACT

Programmers write a ton of code in their daily life. Programmers while coding would like to have some assistance in solving and performing some small scale coding activities where human intelligence is not required. While coding, programmers face issues of not knowing how to do a task or have second thoughts doing it. Usually programmers use internet to solve these issues. After writing all the code programmers have to debug the code for bugs and issues. During this process programmer end up commenting lots of lines of code, writing many print statements and deleting some parts of code. To present their code, programmer's prepare documentation explaining how their code works. We believe that while doing these above tasks programmers waste a decent amount of time.

To solve these challenges, we propose **VATC**, tool that works as a assistant to the programmer, which can be accessed using voice commands, helps the programmer by solving some of the the above mentioned tasks and save a ton of their time.

KEYWORDS

Python, Tkinter, Speech Recognition API, Text Editor, Voice Assistant, Python libraries, Bert model

1 INTRODUCTION

Developing a new Software and Programming is the new Era and the current trend all over the World [1]. The rate of evolution and development of new Software is very high[1]. This Rapid Development in the Software Industry, has lead to more complexity and efficiency in Programs. Due to this, Programming and Software Development has become more important than ever. These changes in the IT Industry had two major outcomes - "more opportunities for new Programmers, very high expectations from Programmers" and "massive number of Libraries and different new Programming techniques", which made Programming tasks difficult than ever. At this rate, it has become more difficult to maintain Efficiency in writing Programs[2]. Due to the increase of Libraries and Programming techniques, finding libraries and code snippets has become more difficult[3][4]. Programmers started developing new libraries for every new task and technique. This made the programmers put more efforts in finding the perfect library for their task. Due to the complexity in the programs, the process of debugging started taking most of the programmer's time[6]. This lead many companies to start working on Text editors that help with few of the above mentioned tasks like debugging, code prediction, code summarizing, to increase the rate of coding process[7][8]. But most of these were not accurate. Few Text editors like Visual Studio Code had specially designed extensions which would make the coding process more simpler. But all these soft-wares were using only one resource from the user - "User's Hands".

Given the popularity and the evolution of programming languages, we propose an idea that uses two resources from the user - "User's Hands" and "User's Voice"[9]. We pivoted from the traditional way of programming, and came up with this revolutionary idea that we believe would be an evolution in the field programming. In the entire process of programming one of the user's most important resource "Voice" is not being used effectively[10]. But with the help of User's Voice and few basic machine learning algorithms we can complete some basic tasks of the coding process like finding code snippets from internet, finding required libraries using few parts of the code[5]. These basic features of the text editor would end up saving a lot of time for the programmer and also make the program more effective and efficient.

We propose VATC, a Text editor that uses both the user's resources to help with basic tasks like finding required libraries, suggesting code snippets from few famous sites like stack over flow, Geeks for Geeks, and help with the process of debugging and documentation. This paper contributes (i) an algorithm for finding relevant libraries, code snippets (ii) an API that takes accurate decisions to edit programs based on voice commands (iii) an API to Document your project by dictating. VATC uses Speech Recognition API that converts programmer's voice into text, and with the help of optimised NLP algorithms, that text is categorised to get the best results. Considering the current trend and the rise of Python language, VATC is designed to work with any kind of python language, and find required libraries and code snippets, all while programming without any delay or discomfort[11].

2 DESIGN OF THE TOOL

Based on our research we found that there are only few editor, that supports NLP and ML features to be a part an extension developed for that editor. To use these ML features in the extension, there has to be a server maintained to perform these actions. we believe maintaining a server and receiving requests would be less responsive and take lot of time to process them. Instead of using a server based extension that allows the developer to add Machine Learning features like NLP, we developed our own Editor using Tkinter to support Machine Learning features along with basic editor features. In this paper we propose the solution for using Machine Learning features in a editor without server.

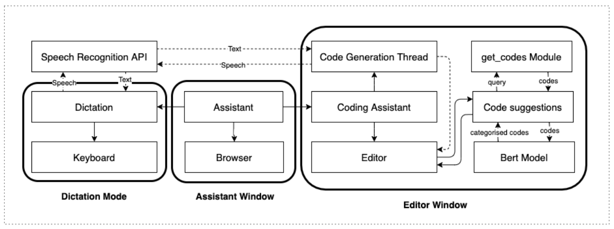


Figure 1: Design of VATC

In this paper we propose an editor that is developed with python tools and libraries, to support inbuilt ML features without the help of servers. Firstly, we designed an editor using tkinter(graphic user interface library) for writing the program. Then using the speech Recognition API 1 we convert the input audio into text, and apply NLP techniques to convert the input text into a meaningful python program. This converted python program can be inserted into the program.

Listing 1: Speech Recognition API to convert audio into text.

```
import speech_recognition as sr
r = sr.Recognizer()


def speech2Text():
    with sr.Microphone() as source:
        audio = r.listen(source)
        try:
            text = r.recognize_google(audio)
            return text
        except:
            return -1
```

To find the required libraries and code snippets from internet, we decided to design a sidebar that takes query as input and gives code snippets and libraries as output. This reference search algorithm takes the input query and converts it into a internet understanding language by removing complex equations, and acquires libraries from internet. These received HTML pages are then scraped using Beautiful Soup library to extract code snippets and libraries from web pages. All these outputs are then passed through a bert model, that summarises all the code snippets and categorise them based on programmer's code to display most relevant snippets on the top. BERT model is developed using k-train python library using 35000 input parameters to get more accuracy and better results. we trained bert model on StackOverFlow and GeeksforGeeks posts dataset that contains code snippet, respective phrases as dependent variables and their scores as independent variables. The scores of each post will help the model to decide how relevant the post is when compared to the programmer's code and the search query, and can also be used to categorise the posts and display the post with highest score on the top. Using this model we ended up with an accuracy of 82% for finding related code snippets and libraries, based on the evaluation dataset.

Listing 2: Algorithm to categorise code snippets.

```
def categorise_posts(sof_codes):
    for i in range(len(sof_codes)):
        votes = sof_codes[i][2]
        com_score = 0
        for j in range(len(sof_codes[i][3])):
            com_sent = ''
            if is_predictor == 1:
                cs = sof_codes[i][3][j][1]
                com_sent = bert_model.predict(cs)
            else:
                com_sent = 'pos'
            if com_sent == 'neg':
                code_score = sof_codes[i][3][j][2]+1
                com_score += (-1*code_score)
            else:
                com_score += (sof_codes[i][3][j][2] + 1)
        score = votes + (com_score/4)
        sof_codes[i].append(score)
    sof_codes.sort(key = lambda x: x[4])
```

As mentioned earlier, we developed a feature that allows the programmer to prepare document for the project just by dictating. Based on the fact that, Documentation will be done after programming and by referring to the program, we decided to include this feature in a different window so that user can simultaneously look at the code in the editor and prepare his documentation by dictating. To implement this feature firstly, we pass the input to the Speech Recognition API to convert it into text. This converted text is then prepossessed to fit perfectly in the document, and then using the keyboard Python library received text will be typed into the editor.

3 FUNCTIONING OF THE TOOL

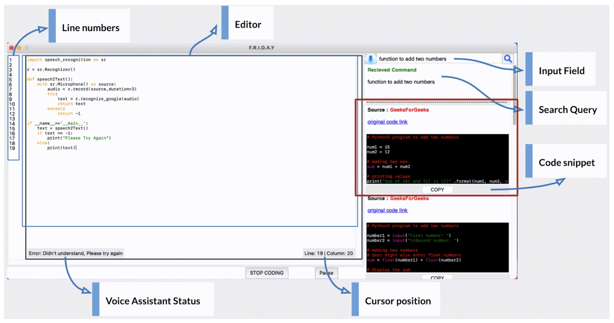
VATC offers functionalities and features - "Python programming editor", "Programming with voice commands", "helping with basic debugging tasks", "Finding related code snippets and required libraries", "Documentation by dictation".

Python programming editor is an editor that is designed for python language to display keywords and syntax in an elegant manner. This editor comes with some in build functionalities like "Programming with voice commands" and "helping with basic debugging tasks". Programming with voice commands functionality can be used to write python program using basic voice commands. Programmer can use this functionality to import libraries and functions, define functions, write print statements etc.

VATC editor also helps programmers with basic debugging tasks like commenting lines, including print statements, uncommenting lines, deleting pieces of code. Finding related code snippets and required libraries functionality can be used to find code snippets and required libraries optimized for the programmer's code just by passing a search query either by voice or input field. The received code snippets can be copied from the sidebar and pasted into the code using a copy button attached to every code snippet. Documentation by dictation functionality can be used to type the document just by dictating to the tool. Considering the fact that most humans speak faster than they type, we decided to collaborate the powerful elements of human voice and computer typing, processing capacity to improve the speed of the documentation process and save time for the programmer.

4 USE CASE

Consider a scenario where the user has to develop software using Python language but only has basic knowledge about python and its libraries. After developing the project, the user has to prepare Documentation for the project describing the project and its features. For the above stated scenario our tool VATC can help to achieve both speed and accuracy in writing the code and documenting the project.



```
import speech_recognition as sr

r = sr.Recognizer()

def speech2Text():
    with sr.Microphone() as source:
        audio = r.record(source,duration=3)
        try:
            text = r.recognize_google(audio)
            return text
        except:
            return -1

if __name__=='__main__':
    text = speech2Text()
    if text == -1:
        print("Please Try Again")
    else:
        print(text)
```

Error: Didn't understand, Please try again

Line: 19 | Column: 20

function to add two numbers

```
# Python3 program to add two numbers

num1 = 15
num2 = 12

# Adding two nos
sum = num1 + num2

# printing values
print("Sum of {0} and {1} is {2}" .format(num1, num2, s
```

```
# Python3 program to add two numbers

number1 = input("First number: ")
number2 = input("\nSecond number: ")

# Adding two numbers
# User might also enter float numbers
sum = float(number1) + float(number2)

# Display the sum
```

Figure 2: VATC Editor fields and features

If the above stated user uses VATC, that user can define functions and call them in the program with voice commands. As the user is new to python programming it would be difficult for the user to find libraries and better techniques, but using the "Finding code snippets and libraries" functionality user can find best results for his code from the best sites and with very high user ratings just using a search query. Because of this feature, there is no need for the user to open a browser and find the required libraries and code snippets. This feature can be used while programming, so that the user can give the search query to the tool and work on some other feature while we search for the code snippets. This feature saves a ton of user's time and also finds the perfect results for the program. After developing the project, user can prepare Documentation for the project using "Documentation using dictation" feature that allows the user to type using dictation. Finally by using VATC tool, user can develop the complete project in an efficient way within less time than expected.

5 LIMITATIONS OF THE TOOL

VATC editor only supports Python language syntax. All the debugging features, programming features support only python language syntax. Only python keywords gets highlighted. Finding code snippets and libraries feature also gives optimised and accurate results only for python language. For finding code snippets from internet we opted only few famous sites due to which we may not consider few code snippets which might end up being perfect fit for the programmer's code. While displaying the code snippets to the user we only considered few features like votes, relation of snippet to the query, to categorise the snippets. To implement the Dictation functionality we make very few changes and optimisation for the user's input command. Programming with voice feature supports only few basic and important python features like defining functions, writing print statements, commenting and uncommenting lines, deleting pieces of code etc.

Finally as we only performed few basic noise reducing techniques, the speech recognition API might not work properly in a very noisy environment. If the tool is used in an noisy environment then it might end up being less accurate and less responsive.

6 RELATED WORK

The related research falls in the field of voice recognition to support computer programming.

John Edward reports that the fundamental technology for continuous voice recognition significantly improved in 1997[12].Like Srinivasan and Vergo [13], our design is based on a commercially available voice recognition development environment. Their work focuses on making the development environment better. We are focusing on making a programming environment that can be controlled by voice alone.

Voice coding isn't as simple as it seems, with layers of complex technology behind it. The voice-coding app Serenade, for instance, has a speech-to-text engine developed specifically for code, unlike Google's speech-to-text API, which is designed for conversational speech. Once a software engineer speaks the code, Serenade's engine feeds that into its natural-language processing layer, whose machine-learning models are trained to identify and translate common programming constructs to syntactically valid code.

Talon, a hands-free coding platform. "The point of Talon is to completely replace the keyboard and mouse for anyone".Talon has several components to it: speech recognition, eye tracking, and noise recognition. Talon's eye tracking and noise-recognition capabilities simulate navigating with a mouse, moving a cursor around the screen based on eye movements and making clicks based on mouth pops. You'll need eye-tracking hardware if you want to run Talon with eye tracking. Open-source voice-coding platforms such as Aenea and Caster are free, but both rely on the Dragon speech-recognition engine, which users will have to purchase themselves.

7 CONCLUSION AND FUTURE WORK

Considering the rise of Software Development and evolution of new programmers, programming has become difficult than ever[1]. New softwares are being developed and new programming techniques are being discovered. This made the programming experience of new programmers more complex. Based on the fact that softwares have become more complex and large, documenting them has also become a time consuming process. To reduce the difficulty in programming, we propose the idea of an editor that uses both voice commands and keyboard seamlessly to code and debug the code. Along with that, VATC helps the programmer to find code snippets and libraries suitable for his program from some of the best websites to save the time of the programmer by avoiding the traditional way, which is time consuming and might be less accurate for new programmers. VATC also has a feature called Documentation by dictation, that allows the user to document their project just using their voice. we believe that programmer's would find this tool helpful for their programming and software development tasks, because it saves a lot of time for the programmer and also makes the programming process much easier.

we plan to extend this tool to work in noisy environments by improving the speech recognition API, so that the tool can detect programmer's voice more accurately and work without any discomfort. VATC can be improved to search more popular websites for code snippets and libraries. The search algorithm for searching code snippets and libraries, can be improved by including more factors like summarising the code and using that as a dependent variable. we plan to improve the Bert model by training the model on code snippets and their summary, to improve the search results of the tool. VATC can also be extended to support other programming languages like JavaScript, Java, and HTML. we also plan to improve the tool to suggest APIs based on the program to reduce the footprint and improve quality of the code.

8 ARTIFACTS

GitHub link of the tool.
https://github.com/Gamemaster-007/VATC

REFERENCES

[1] Paula Hunter, Stephen Walli , *"The Rise and Evolution of the Open Source Software Foundation". IFOSS L. Rev. 31 (2013)*

[2] Dan Tian , *"Evaluation Model and Empirical Study for Innovation Efficiency of Software Industry". 10.1109/ICDMA (2013)*

[3] Mashal Ahmad, Mel Ó Cinnéide , *"Impact of Stack Overflow Code Snippets on Software Cohesion: A Preliminary Study". 10.1109/MSR (2019)*

[4] Agnieszka Ciborowska, Nicholas A. Kraft, Kostadin Damevski , *"Detecting and Characterizing Developer Behavior Following Opportunistic Reuse of Code Snippets from the Web". IEEE/ACM (2018)*

[5] Lei Ai, Zhiqiu Huang, Weiwei Li, Yu Zhou, Yaoshen Yu , *"SENSORY: Leveraging Code Statement Sequence Information for Code Snippets Recommendation". 10.1109/COMPSAC (2019)*

[6] Gao Gao, Ashley Hale, Michelle Ichinco , *"Designing a Support Tool for API Debugging". 10.1109/VLHCC (2019)*

[7] Rajesh Prabhu, Ninad Phutane, Shiv Dhar, Siddhesh Doiphode , *"Dynamic formatting of source code in editors". 10.1109/ICIIECS (2017)*

[8] Kaiz Merchant, Yash Pande , *"NLP Based Latent Semantic Analysis for Legal Text Summarization". 10.1109/ICACCI (2018)*

[9] A.S. Elmaghraby , *"Voice recognition applications for programming environments". 10.1109/SECON (1989)*

[10] Jungyoon Choi, Haeyoung Gill, Soobin Ou, Yoojeong Song, Jongwoo Lee , *"Design of Voice to Text Conversion and Management Program Based on Google Cloud Speech API". 10.1109/CSCI46756 (2018)*

[11] S Subhash, Prajwal N Srivatsa, S Siddesh, A Ullas, B Santhosh , *"Artificial Intelligence-based Voice Assistant". 10.1109/WorldS450073 (2020)*

[12] John Edwards , *"Voice Based Interfaces make better PC Listeners, Computer". August (1977)*

[13] Savitha Srinivasan and John Vergo , *"Object Oriented Reuse: Experience in Developing a Framework for Speech Recognition Applications, The 20th International Conference on Software Engineering, Kyoto, Japan". pp 322-330, 19 - 25 April (1988)*