SPEECH2CODE: Speech to Code Converter Using AI

Soham S. Kerimane[1], Sufiyan U. Patel[2], Parshwa S. Navale[3], and Pushpender Sarao[4]

[1] Student, Computer Science and Engineering, Sharad Institute of Technology College of Engineering, Maharashtra, India
[2] Student, Computer Science and Engineering, Sharad Institute of Technology College of Engineering, Maharashtra, India
[3] Student, Computer Science and Engineering, Sharad Institute of Technology College of Engineering, Maharashtra, India
[4] Head of Department , Computer Science and Engineering, Sharad Institute of Technology College of Engineering, Maharashtra, India

ABSTRACT

Coding or programming plays an important role in a programmer's life and there are several ways we can minimize the time it take to complete a code. Such as code editors provide various suggestions by seeing the starting letter of the coding, there are several templates available on specific problem statements like searching algorithm, sorting algorithm, etc. Automation is the technique to save the time as well as the cost of a program. Using Artificial Intelligence we can generate such solutions which can solve the coding problems with the ideal origination

Keyword : *Automation ,code editors, templates, Artificial Intelligence, searching algorithm, sorting algorithm*

1. INTRODUCTION

Coding or programming plays an important role in a programmer's life. And there are several ways we can minimize the time it take to complete a code. such as code editors provide various suggestions by seeing the starting letter of the coding, there are several templates available on a specific problem statements like searching algorithm, sorting algorithm, etc. although it is true that we need some shortcuts to save the time in the coding, we have to make sure the content as well as the quality of our code is better. And in some cases we need automation in our coding like writing same code again and again, using same code block in different persona. In such cases it is better to automate these things to save the valuable time. So in this project we are going to use the automation to the completion of the code as our main solution. Using natural language processing and machine learning we are going to find the feasible solution for our problem statement.

In a big project, we need to write code in a more familiar way that anyone from our colleagues can read it and understand it ideally. As long as we take care about the time, we have to make sure that it should take less time and should contain code that is readable. Using automation we can achieve these types of goals by using machine learning for understanding the problem statement and natural language processing for generating the specific instructions

2. LITERATURE SURVEY

GitHub Copilot is an AI pair programmer. GitHub Copilot is powered by a new AI system developed by OpenAI Codex and is coming soon to Visual Studio Code. It aims to help Programmers code faster. It basically draws context from the code you’re working on, suggesting whole lines or entire functions. OpenAI Codex has a broad knowledge of how people use code and is significantly more capable than GPT-3 in code generation.

It can suggest complete lines of code or entire functions by analyzing how you code. GitHub Copilot can assemble code from user comments and predicts your code by just reading the function name you have declared. It allows you to cycle through alternative suggestions and manually edit the suggested code. It autofill repetitive code, or create unit tests for your methods.

The GitHub Copilot editor extension sends your comments and code to the GitHub Copilot service, which then uses OpenAI Codex to synthesize and suggest code. it actually works by reading through all the open-source code on the GitHub repos worldwide and then collect the data and tries to find the best possible code related to it! It is said to work great with repetitive code patterns so users can let it generate the rest of the code. The AI assistant can also help you learn a new programming language.

It is said to have been tested against a set of Python functions that have good test coverage in open source repos by blanking out the function bodies and asked GitHub Copilot to fill them in. The model got this right 43% of the time on the first try, and 57% of the time when allowed 10 attempts. And it's getting smarter all the time.

GitHub Copilot tries to understand your intent and to generate the best code it can, but the code it suggests may not always work or even make sense. "GitHub Copilot draws context from the code you're working on, suggesting whole lines or entire functions," GitHub CEO Nat Friedman explained in a blog post introducing the technology. The algorithm consistently improves by recording whether each suggestion is accepted or not.

The GitHub Copilot editor extension sends your comments and code to the GitHub Copilot service, which then uses OpenAI Codex to synthesize and suggest code. it actually works by reading through all the open-source code on the GitHub repos worldwide and then collect the data and tries to find the best possible code related to it! It is said to work great with repetitive code patterns so users can let it generate the rest of the code. The AI assistant can also help you learn a new programming language.

It is said to have been tested against a set of Python functions that have good test coverage in open source repos by blanking out the function bodies and asked GitHub Copilot to fill them in. The model got this right 43% of the time on the first try, and 57% of the time when allowed 10 attempts. And it's getting smarter all the time.

In order to make the most out of it, it is suggested to divide the code into smaller functions, provide meaningful function names, parameters, and docstrings.

3. METHODOLOGY

- Data Collection
- Data Pre-processing
- Converting to Categorical Data
- Training the Model
- Deployment of Model

3.1 Data Collection:

We are generating the data on our own. We will write a python script that will generate tons of statements and then store it in a data frame. this dataframe contains multiple problem statement with their respective lables.

Eg.

Problem Statement	**Lable**
write a function that can take 2 integers as the input	[function, 2, integers]
write an if else condition	[if-else condition]

3.2 Data Pre-processing:

Before training the model we have to convert the data into categorical format. We will use labelencoder() a SKlearn library for that purpose. in data pre-processing we will mainly convert the text data into categorical format.

we have used tensorflow tokenizer to tokenize the text and convert the string data into categorical format, converted this tokenized data into sequences of integers. so we will get an array of some integers. finally we will add some padding to normalize the array to a specific shape using tensorflow pad sequences.

for lables we have used onehotencoder() liberary to convert the data into categorical format as it contains multiple datatypes, onehotencoder() gives the best performance amoung all the liberaries.

Training Model:

After getting all the data in a well good manner we will split the data into training and validation sets using sklearn train_test_split() liberary by giving the 10% as the validation set.

We will use Tensorflow and keras for training the Neural Network. Then we will convert that model into a javascriipt object notation format (JSON).

we need to use various libraries from tensorflow and keras such as modles, layers, intitializers and optimizers.

1. Model- Keras.models.Sequential

2. Layer-

   keras.layers.Embedding

   keras.layers.LSTM

   keras.layers.Dense

   keras.layers.Dropout

3. Initializer - keras.initializer.Constant

4. Optimizer - keras.optimizer.Adam

3.3 Deploying Model:

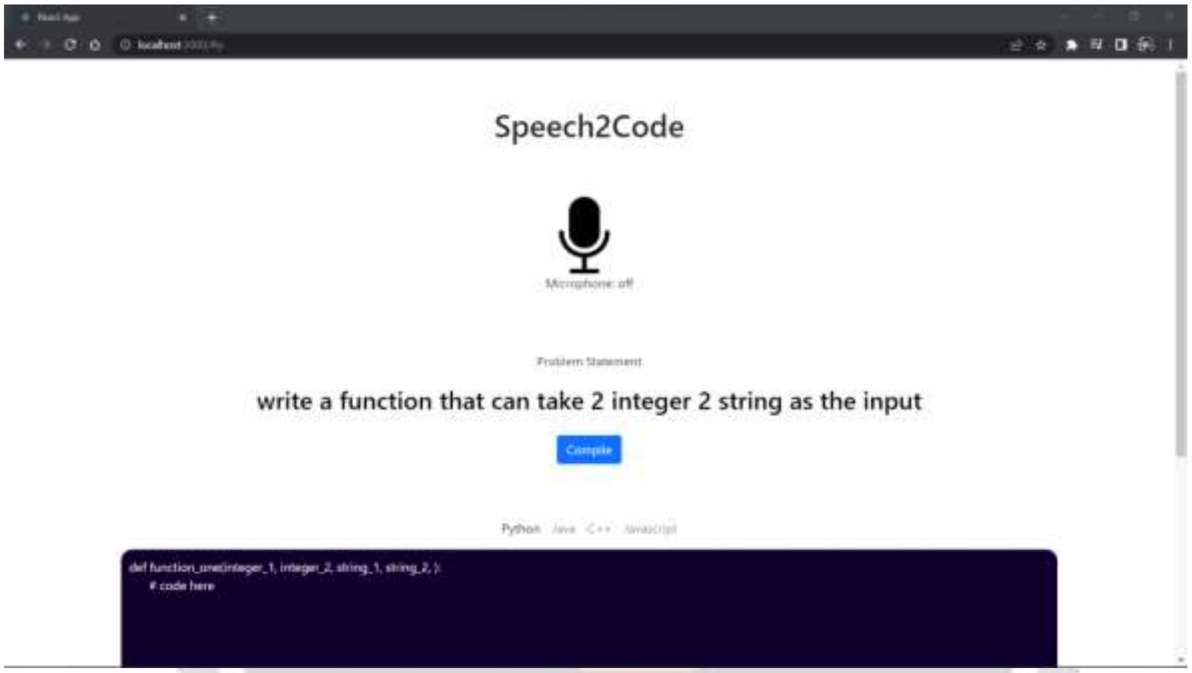
We will upload the trained model to a server and then access it from our web-app. So anyone with this web-app can use this technology globally.

In our case we will upload the model on github and access that model in react js using tensorflow js library.

4. RESULTS AND DISCUSSION:

here's the final output of our project, a web app that can take problem statement in speech format and will provide the solution code in the code snippet.

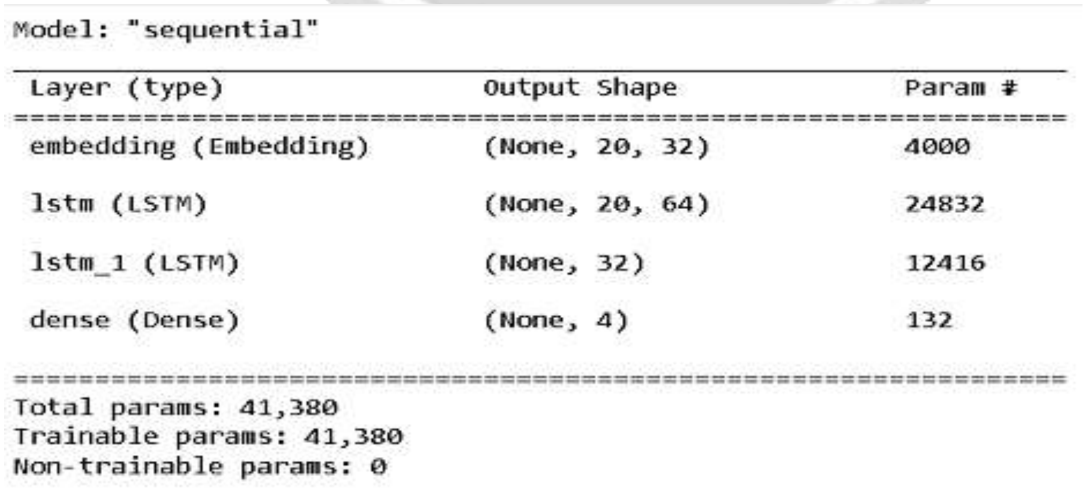
we have to press the microphone button, it will take our speech as the problem statement or we can just type it in the input box. then after clicking on the compile button the system will understand the problem and according to the selected programming language it will generate the solution.



write a function that can take 2 integer 2 string as the input

```
def function_one(integer_1, integer_2, string_1, string_2, ):
    # code here
```

Neural Networks:



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 20, 32)            4000

 lstm (LSTM)                 (None, 20, 64)            24832

 lstm_1 (LSTM)               (None, 32)                12416

 dense (Dense)               (None, 4)                 132

=================================================================
Total params: 41,380
Trainable params: 41,380
Non-trainable params: 0
```

5. CONCLUSION:

Thus, this project will give acceleration to developers who want to write code in a more advanced way. it will save their time and efforts that they used to spend on written same as well as boring code again and again. As a matter of fact when people use this technology, we will get more data from their code and we can improve our model's performance using the new data recursively.

6. REFERENCES:

[1] Cwe-327: Use of a broken or risky cryptographic algorithm, 2006.

URL https://cwe.mitre.org/data/definitions/327.html.

[2] Cwe-780: Use of rsa algorithm without oaep, 2009. URL https:

//cwe.mitre.org/data/definitions/780.html.

[3] A6:2017-security misconfiguration, 2017. URL https:

//owasp.org/www-project-top-ten/2017/

A6 2017-Security Misconfiguration.html.

[4] Abid, A., Farooqi, M., and Zou, J. Persistent anti-muslim bias in

large language models. arXiv preprint arXiv:2101.05783, 2021.

[5] Acemoglu, D. and Restrepo, P. Robots and jobs: Evidence from us

labor markets. Journal of Political Economy, 128(6):2188–2244,

2020a.

[6] Acemoglu, D. and Restrepo, P. The wrong kind of ai? artificial intelligence and the future of labour demand. Cambridge Journal

of Regions, Economy and Society, 13(1):25–35, 2020b.

[7] Agrawal, H., Horgan, J. R., London, S., and Wong, W. E. Fault

localization using execution slices and dataflow tests. Proceedings of Sixth International Symposium on Software Reliability

Engineering. ISSRE'95, pp. 143–151, 1995.

[8] Allamanis, M., Tarlow, D., Gordon, A., and Wei, Y. Bimodal modelling of source code and natural language. In Bach, F. and Blei,

D. (eds.), Proceedings of the 32nd International Conference

on Machine Learning, volume 37 of Proceedings of Machine

[9] Learning Research, pp. 2123–2132, Lille, France, 07–09 Jul

2015. PMLR. URL http://proceedings.mlr.press/

v37/allamanis15.html.

[10] Alley, E. C., Khimulya, G., Biswas, S., AlQuraishi, M., and Church, G. M. Unified rational protein engineering with sequence-based deep representation learning. Nature methods, 16(12):1315–1322, 2019.