

PROJECT Finalization REPORT ON Self Learning Bot

Student Details

<u>Roll No</u>	<u>Name</u>
20201CAI0104	Maheswar Reddy
20201CAI0131	Karthik Sharma
20201CAI0152	Arun Teja
20201CAI0096	Salapakshi Ganesh

Under the Supervision of :

Prof . Dr.Asif Mohammed H.B

October , 2023

**DEPARTMENT OF COMPUTER SCIENCE AND
ENGINEERING
PRESIDENCY UNIVERSITY
BENGALURU**

Abstract:

In this project, we propose the development of a sophisticated chatbot equipped with self-learning capabilities, modern Natural Language Processing (NLP) techniques, and advanced cognitive functionalities. The chatbot aims to intelligently adapt its responses over time, catering to users with varying levels of technical expertise. Key features include dynamic learning, answer relevancy scoring, and personalized interactions.

Objective:

The primary objective is to create a chatbot that goes beyond conventional rule-based systems by incorporating self-learning mechanisms. The chatbot will be designed to understand and respond to user queries in a manner that adapts to the user's technical proficiency, ensures answer relevancy, and continuously improves through interaction.

Sample_Code:

```
import re

import longResponse as long

def message_probability(user_message,
                        recognised_words, single_response=False,
                        required_words=[]):
    message_certainty = 0
```

```

has_required_words = True
for word in user_message:
    if word in recognised_words:
        message_certainty += 1
        percentage = float(message_certainty) /
float(len(recognised_words))
        for word in required_words:
            if word not in user_message:
                has_required_words = False
                break
if has_required_words or single_response:
    return int(percentage * 100)
else:
    return 0

def check_all_messages(message):
    highest_prob_list = { }

    def response(bot_response, list_of_words,
single_response=False, required_words=[]):
        nonlocal highest_prob_list

        highest_prob_list[bot_response] =
message_probability(message, list_of_words,
single_response, required_words)

```

```

    response('Hello!', ['hello', 'hi', 'hey', 'sup', 'heyo'],
single_response=True)

    response('See you!', ['bye', 'goodbye'],
single_response=True)

    response('I\'m doing fine, and you?', ['how', 'are',
'you', 'doing'], required_words=['how'])

    response('You\'re welcome!', ['thank', 'thanks'],
single_response=True)

    response('Thank you!', ['i', 'love', 'code', 'palace'],
required_words=['code', 'palace'])

    response(long.R_ADVICE, ['give', 'advice'],
required_words=['advice'])

    response(long.R_EATING, ['what', 'you', 'eat'],
required_words=['you', 'eat'])

    best_match = max(highest_prob_list,
key=highest_prob_list.get)

    print(highest_prob_list)

    return long.unknown()

if highest_prob_list[best_match] < 1 else best_match

def get_response(user_input):

    split_message = re.split(r'\s+|[,;?!.-]\s*',
user_input.lower())

    response = check_all_messages(split_message)

```

return response

while True:

print('Bot: ' + get_response(input('You: ')))

longResponses.py:

import random

R_EATING = "I don't like eating anything because I'm a bot obviously!"

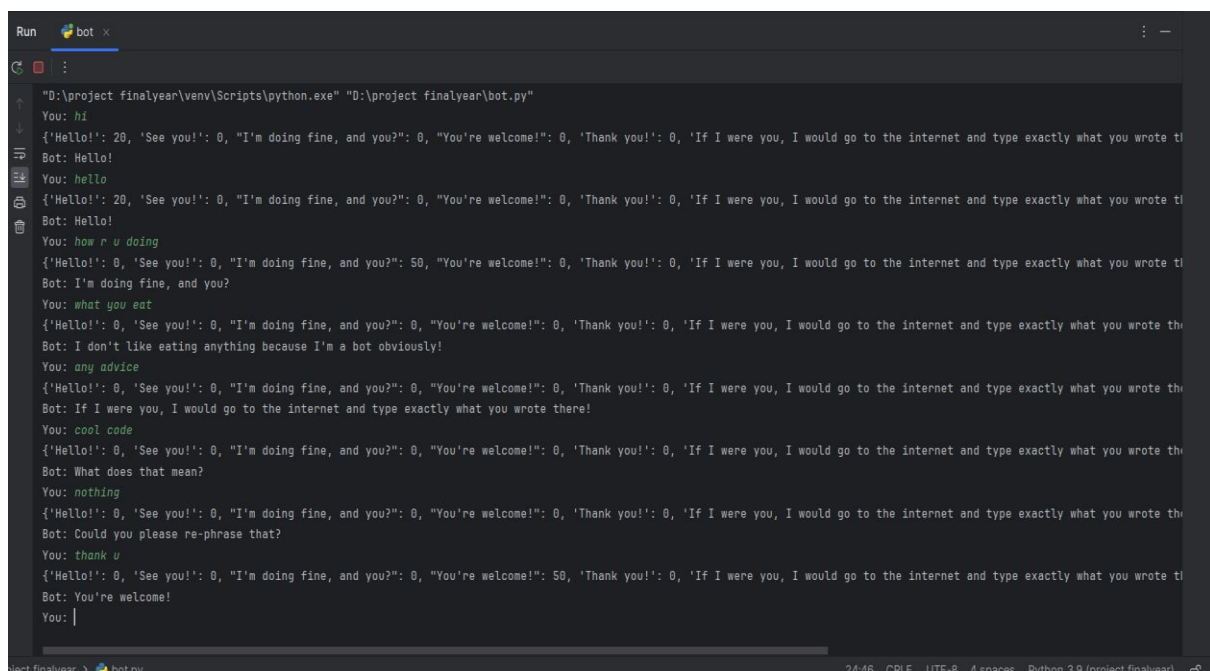
R_ADVICE = "If I were you, I would go to the internet and type exactly what you wrote there!"

def unknown():

response = ["Could you please re-phrase that? ", "...",
"Sounds about right.", "What does that mean?"]
[random.randrange(4)]

return response

Output:



```
Run bot x
D:\project finalyear\venv\Scripts\python.exe "D:\project finalyear\bot.py"
You: hi
{'Hello!': 0, 'See you!': 0, 'I'm doing fine, and you?': 0, 'You're welcome!': 0, 'Thank you!': 0, 'If I were you, I would go to the internet and type exactly what you wrote there!': 0}
Bot: Hello!
You: hello
{'Hello!': 20, 'See you!': 0, 'I'm doing fine, and you?': 0, 'You're welcome!': 0, 'Thank you!': 0, 'If I were you, I would go to the internet and type exactly what you wrote there!': 0}
Bot: Hello!
You: how r u doing
{'Hello!': 0, 'See you!': 0, 'I'm doing fine, and you?': 50, 'You're welcome!': 0, 'Thank you!': 0, 'If I were you, I would go to the internet and type exactly what you wrote there!': 0}
Bot: I'm doing fine, and you?
You: what you eat
{'Hello!': 0, 'See you!': 0, 'I'm doing fine, and you?': 0, 'You're welcome!': 0, 'Thank you!': 0, 'If I were you, I would go to the internet and type exactly what you wrote there!': 0}
Bot: I don't like eating anything because I'm a bot obviously!
You: any advice
{'Hello!': 0, 'See you!': 0, 'I'm doing fine, and you?': 0, 'You're welcome!': 0, 'Thank you!': 0, 'If I were you, I would go to the internet and type exactly what you wrote there!': 0}
Bot: If I were you, I would go to the internet and type exactly what you wrote there!
You: cool code
{'Hello!': 0, 'See you!': 0, 'I'm doing fine, and you?': 0, 'You're welcome!': 0, 'Thank you!': 0, 'If I were you, I would go to the internet and type exactly what you wrote there!': 0}
Bot: What does that mean?
You: nothing
{'Hello!': 0, 'See you!': 0, 'I'm doing fine, and you?': 0, 'You're welcome!': 0, 'Thank you!': 0, 'If I were you, I would go to the internet and type exactly what you wrote there!': 0}
Bot: Could you please re-phrase that?
You: thank u
{'Hello!': 0, 'See you!': 0, 'I'm doing fine, and you?': 0, 'You're welcome!': 50, 'Thank you!': 0, 'If I were you, I would go to the internet and type exactly what you wrote there!': 0}
Bot: You're welcome!
You: 
```

Methodology:

1. Data Collection and Preprocessing:

- Gather a diverse dataset of conversations or text data, including technical and non-technical topics.
- Preprocess the data, including tokenization, stemming, and removing noise.

2. NLP and Deep Learning Models:

- Use state-of-the-art NLP models, such as Transformer-based models (e.g., BERT, GPT-3), for understanding and generating text.
- Fine-tune these models on your specific dataset to improve their performance for your chatbot's context.

3. Transliteration Capability:

- Implement or integrate a transliteration module if your chatbot needs to handle multiple languages or scripts.
- Use libraries or services that can transliterate text between different writing systems.

4. User Profiling:

- Develop a user profiling system that assesses the technical ability of users based on their interactions and responses to specific questions.
- Maintain user profiles that include information about their technical knowledge.

5. Answer Relevancy Scoring:

- Implement a mechanism to score the relevancy of answers over time.
- Utilize user feedback and interactions to continuously improve the scoring algorithm.
- Apply techniques like reinforcement learning to optimize answer selection.

6. Response Tailoring:

- Based on the user's technical profile, create rules or models that determine whether to provide a technical or non-technical response.
- Segment your user base into different categories (e.g., novice, intermediate, expert) and tailor responses accordingly.

7. Continuous Learning:

- Enable the chatbot to learn from each interaction and update its models accordingly.
- Implement mechanisms for feedback collection and model retraining.

8. Deployment

- Deploy the chatbot on your preferred platform, whether it's a website, messaging app, or custom application.
- Ensure scalability and reliability for handling user interactions.

9. Monitoring and Evaluation

- Continuously monitor the chatbot's performance, including answer relevancy and user satisfaction.

- Use appropriate evaluation metrics to assess its effectiveness.

10. User Feedback Loop

- Encourage users to provide feedback on the bot's responses and use this feedback for improvements.

11. Security and Privacy:

- Implement security measures to protect user data and interactions.
- Comply with data privacy regulations (e.g., GDPR) as applicable.

12. Integration:

- Integrate the chatbot with external systems, databases, or APIs if it needs to access specific information or perform tasks.

13. Documentation and Support:

- Provide clear documentation on how to interact with the chatbot.
- Offer user support for troubleshooting and assistance.

Conclusion:

The development of a self-learning chatbot with advanced NLP capabilities presents an innovative approach to natural language interactions. The integration of user profiling, answer scoring, and dynamic response tailoring aims to create a more intelligent and user-friendly conversational agent.