Jarvis-music recommendation bot

Dhruv Sharma-11023210198

Akshat Jain-11023210142

Pulkit Pal-11023210173

Tanish Awasthi-11023210197

Index

Introduction

About Jarvis

Libraries used

Codes and output

Conclusion

The power of Jarvis

Jarvis is an AI-powered music recommendation bot designed to provide personalized music suggestions based on users' preferences.

1. Natural Language Processing (NLP): Jarvis understands text input.

2. Music Database: Access to millions of songs from various genres and artists.

3. Personalization: Learns users' preferences and adapts recommendations.

4. Discovery Feature: Introduces users to new artists and genres.

Libraries:

The Requests library is the de facto standard for making HTTP requests in Python. It abstracts the complexities of making requests behind a beautiful, simple API so that you can focus on interacting with services and consuming data in your application.

1. TextBlob is a Python library for processing textual data. It provides a simple API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more. TextBlob stands on the giant shoulders of NLTK and pattern, and plays nicely with both.

Textblob

Pytube

Pytube is a lightweight, Python library used for downloading videos from YouTube. It provides a simple way to interact with YouTube's video streaming platform and enables downloading videos, audio, and metadata like video titles and descriptions directly from YouTube.

NLTK

NLTK (Natural Language Toolkit) is a powerful, open-source Python library used extensively for natural language processing (NLP). It provides a wide range of tools for working with human language data, including tokenization, parsing, classification, stemming, tagging, and more. NLTK is widely used for tasks like text analysis, sentiment analysis, and building language models.

The webbrowser library in Python is a standard library module that provides a simple way to control and interact with web browsers directly from Python code. This library allows you to open URLs in the default web browser and supports multiple browser types.

WEBBROWSER

FLASK

Flask is a lightweight, flexible web framework for Python that is widely used to create web applications, APIs, and microservices. Known for its simplicity and ease of use, Flask provides a basic structure for developing web applications without the additional overhead of more full-featured frameworks like Django.

INPUT

```
#html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Jarvis - Music Recommendation Bot</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>
    <div class="container">
        <h1>Jarvis - Music Recommendation Bot</h1>
        <form method="POST">
            <input type="text" name="user_input" placeholder="Enter your text here..." required>
            <button type="submit">Submit</button>
        </form>
        <div class="message">
            <p>{{ message }}</p>
        </div>
    </div>
</body>
</html>
```

```css
body {
    font-family: Arial, sans-serif;
    background-color: #e0f7fa;
    margin: 0;
    padding: 0;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
}

.container {
    background-color: #ffffff;
    padding: 30px;
    border-radius: 10px;
    box-shadow: 0 4px 15px rgba(0, 0, 0, 0.2);
    max-width: 500px;
    width: 100%;
    text-align: center;
    transition: all 0.3s ease-in-out;
}

.container:hover {
    transform: translateY(-5px);
}

h1 {
    margin-bottom: 20px;
    color: #00796b;
}

form {
    display: flex;
    flex-direction: column;
    gap: 15px;
}
```

```css
input[type="text"] {
    padding: 10px;
    border: 1px solid #ccc;
    border-radius: 5px;
    font-size: 16px;
}

button {
    padding: 10px;
    border: none;
    background-color: #00796b;
    color: white;
    border-radius: 5px;
    cursor: pointer;
    font-size: 16px;
    transition: background-color 0.3s ease;
}

button:hover {
    background-color: #004d40;
}

.message {
    margin-top: 20px;
    color: #555;
}
```

```python
     from textblob import TextBlob
 4   import webbrowser
 5   from pytube import Search
 6
 7   app = Flask(__name__)
 8
 9   def analyze_mood(text):
10       blob = TextBlob(text)
11       sentiment_polarity = blob.sentiment.polarity
12
13       if sentiment_polarity > 0.5:
14           return "excited"
15       elif sentiment_polarity > 0.2 and sentiment_polarity <= 0.5:
16           return "happy"
17       elif sentiment_polarity > 0 and sentiment_polarity <= 0.2:
18           return "content"
19       elif sentiment_polarity < 0 and sentiment_polarity >= -0.2:
20           return "sad"
21       elif sentiment_polarity < -0.2 and sentiment_polarity >= -0.5:
22           return "depressed"
23       elif sentiment_polarity < -0.5:
24           return "angry"
```

main.py
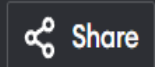
```python
27
28   def recommend_music(mood, music_type, language, attempt=0,
         preference=None):
29       if preference:
30           search_query = preference
31       else:
32           search_query = f"{music_type} songs in {language} for when
               you are feeling {mood}"
33       search = Search(search_query)
34       video_url = search.results[attempt].watch_url
35       return video_url
26
```

```python
40      if request.method == "POST":
41          user_input = request.form["user_input"]
42          print(f"User input: {user_input}")
43          if not hasattr(index, 'mood'):
44              index.mood = analyze_mood(user_input)
45              print(f"Analyzed mood: {index.mood}")
46              return render_template("index.html", message=f"You seem
                    to be feeling {index.mood}. What type of music would
                    you like to listen to (e.g., pop, rock, classical)?"
                )
47          elif not hasattr(index, 'music_type'):
48              index.music_type = user_input
49              print(f"Selected music type: {index.music_type}")
50              return render_template("index.html", message="Great! Now
                    , what language do you prefer for your music?")
51          elif not hasattr(index, 'language'):
52              index.language = user_input
```

```python
55              print(f"Selected language: {index.language}")
56              return redirect(url_for("recommendation"))
57          elif hasattr(index, 'waiting_for_feedback') and index
                .waiting_for_feedback:
58              index.waiting_for_feedback = False
59              if user_input.lower() in ["no", "nah", "nope"]:
60                  print("User did not like the recommendation")
61                  index.attempt += 1   # Try the next recommendation
62                  return redirect(url_for("recommendation"))
63              else:
64                  print("User liked the recommendation")
65                  delattr(index, 'mood')
66                  delattr(index, 'music_type')
67                  delattr(index, 'language')
68                  return render_template("index.html", message="Great!
                        Enjoy your music.")
69          elif hasattr(index, 'waiting_for_preference') and index
                .waiting_for_preference:
70              index.waiting_for_preference = False
71              index.preference = user_input
72              index.attempt = 0
73              print(f"User preference: {index.preference}")
74              return redirect(url_for("recommendation"))
75      return render_template("index.html", message="Hi, I am Jarvis. I
            am your music assistant and I will help you find the perfect
            music. How are you feeling today?")
76
```

```python
77  @app.route("/recommendation")
78  def recommendation():
79      try:
80          recommendation = recommend_music(index.mood, index
                    .music_type, index.language, index.attempt, index
                    .preference)
81          webbrowser.open(recommendation)
82          index.waiting_for_feedback = True
83          print(f"Music recommendation: {recommendation}")
84          return render_template("index.html", message=f"Here's some
                    {index.language} music for you: {recommendation}. Did
                    you like it?")
85      except IndexError:
86          # If no more recommendations are available
87          return render_template("index.html", message="I'm out of
                    suggestions for now. Would you like to tell me your
```