20/12/2024

AI-Based University Navigation System in Python

REPORT INDEX

Report on AI-Based Navigation System Using Speech Recognition and Text-to-Speech

Introduction

This project aims to develop an AI-powered navigation system that can assist users in navigating through a campus. The system uses speech recognition to accept natural language queries and provides spoken directions using text-to-speech (TTS) synthesis. The system utilizes a graph representation of the campus, where nodes represent key locations and edges represent the connections between them. The user can ask for directions between two specific locations, and the system will respond with the path details both in text and voice.

Key Components:

- **Speech Recognition:**

  The system uses SpeechRecognition to process voice inputs from the user. It converts spoken queries into text to extract the start and destination locations.

- **Natural Language Processing (NLP):**

  The spaCy library is employed to understand the text input from the user, identifying and extracting the start and destination locations by matching them with predefined location names in the system.

- **Graph Representation:**

  The campus layout is represented as a graph, where each location is a node, and edges between the nodes represent possible paths between locations.

- **Text-to-Speech (TTS):**

  The system uses pyttsx3 for text-to-speech synthesis, allowing it to provide spoken directions to the user.

- **AI Pathfinding:**

  The system uses a depth-first search (DFS) algorithm to find all possible paths between the start and destination nodes, ensuring that the system can provide users with different route options.

Project Components and Techniques:

1. **Speech Recognition:**

- Library Used: SpeechRecognition
- Functionality:

  The system listens for a user's speech using the microphone and converts it into text. This is essential for users to interact with the system in a hands-free manner.

- Implementation Details:

  The speech_recognition.Recognizer class listens to the user and transcribes the speech using Google's Web Speech API.

- Challenges Handled:

  The project handles cases like unrecognized speech or network issues by providing appropriate feedback to the user.

2. **Natural Language Processing (NLP):**

- Library Used:

  spaCy

- Functionality:

  NLP is used to extract key locations (start and end points) from the user's natural language query. spaCy is employed to parse the query and identify named entities or proper nouns that match the predefined locations in the graph.

- Implementation Details:

  spaCy is used to process the text query and compare the extracted entities against the list of node names (locations) in the campus. The program matches the named locations to the graph’s nodes.

- Challenges Handled:

  The system handles location name variations, such as different formats of names (e.g., full or shortened names).

3. **Graph Representation:**

- Data Structure Used:

  A directed graph is used to represent the campus layout, where each node corresponds to a location (e.g., “Main gate,” “Tennis Court”), and edges represent pathways between these locations.

- Functionality:

  The graph enables pathfinding from one location to another, where the system determines which nodes are reachable from any given start point.

- Implementation Details:

  The graph is represented using a dictionary, where the keys are location identifiers (e.g., "A", "B", "C") and the values are dictionaries of neighboring locations with path costs (weights).

4. **Pathfinding Algorithm:**

- Algorithm Used: Depth-First Search (DFS)
- Functionality: DFS is used to explore all possible paths between the start and destination locations. The system uses recursion to explore all neighbors of a node until it reaches the destination or exhausts all options.
- Implementation Details: The system iterates through all possible paths and collects the ones that lead to the destination. Multiple paths are stored and presented to the user.
- Challenges Handled: The system efficiently handles cases where there are multiple paths, ensuring all possible routes are considered.

5. **Text-to-Speech (TTS):**

- Library Used:

  pyttsx3

- Functionality:

  Once the path is determined, the system uses TTS to speak the directions to the user. It converts the system's output into audio, making the navigation system accessible and interactive.

- Implementation Details:

  The pyttsx3 library is initialized to create a speech engine. The engine converts text into speech and outputs it through the system's speakers.

- Challenges Handled:

  The system provides clear and easily understandable spoken directions, ensuring the user can follow them accurately.

Libraries Used in Python

This project utilizes several key Python libraries to handle speech recognition, text-to-speech conversion, natural language processing, and graph traversal.

1. **SpeechRecognition (sr)**: Used to convert speech to text. It captures audio input from the user and transcribes it into a query for processing.
- **Key Functions**: recognize_google() for speech-to-text, listen() for capturing audio.
2. **pyttsx3**: Converts text to speech, providing audio feedback to the user, such as reading out the navigation paths.
- **Key Functions**: say() for speaking text, runAndWait() for processing speech.
3. **spaCy**: A natural language processing library used to extract relevant locations from the user's spoken query, allowing the system to understand start and end points.

- **Key Functions**: spacy.load() for NLP models, lower() for case-insensitive location matching.

These libraries work together to provide voice-based interaction, natural language processing, and efficient graph traversal for the campus navigation system.

Type of Search Used

Starting at the source node, DFS explores each adjacent node recursively. If a node has not been visited, DFS moves deeper into that node's adjacent nodes. When a destination node is found, the path is recorded.

In the context of this campus navigation system, various search techniques are employed to effectively find the optimal paths from a starting location to a destination. The primary search method used in this project is a **Depth-First Search (DFS)** algorithm. Below are the types of searches used and their role in the system:

Depth-First Search (DFS):

DFS is the core search technique used to find all possible paths between the start and

destination nodes. It is a **graph traversal** method where the algorithm explores as far as possible along each branch before backtracking. Here's how it works in this project.

Types of Graph Used

The campus is represented as a **graph** in this system, where locations (nodes) are connected by paths (edges). The graph is modeled as an **undirected, weighted graph** where the nodes represent physical locations on the campus, and the edges represent the paths between these locations, with weights corresponding to travel distances or times.

1. Undirected Graph:

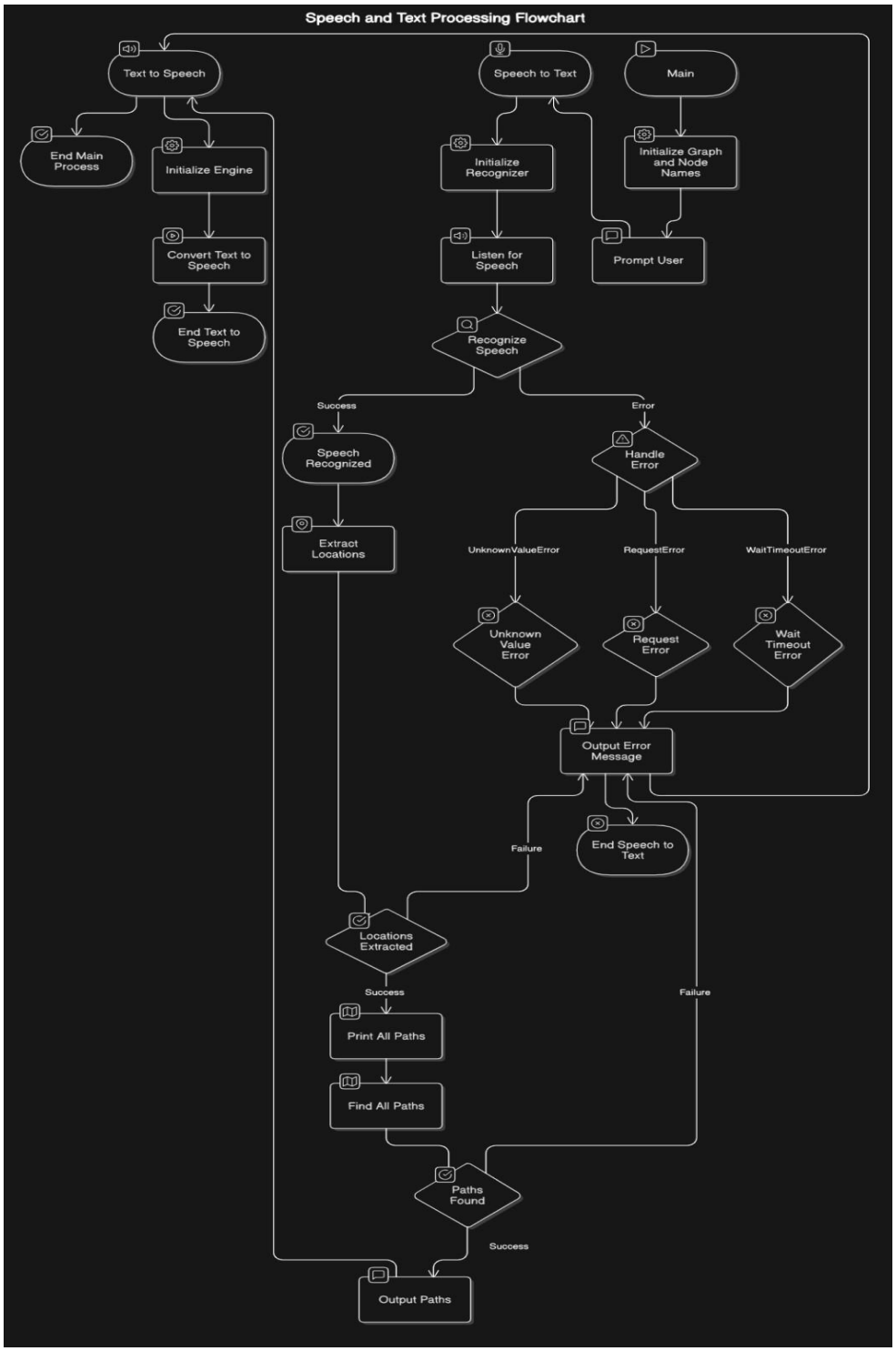
The graph in this project is **undirected**, meaning the paths between locations are bidirectional. For instance, if there is a path between "Main gate" and "Tuc Shop," the reverse path is also valid, indicating that one can travel in both directions.

- **Example**:
  - The path from "Main gate" to "Tuc Shop" has the same weight as the path from "Tuc Shop" to "Main gate."

Project Workflow:

1. **User Input:**
   - The user speaks a query into the microphone, asking for directions between two locations on campus (e.g., "How can I get from Main gate to CS Department?").
2. **Speech-to-Text:**
   - The speech recognition system processes the input and converts the spoken words into text. If speech recognition fails (e.g., unclear speech or network issue), an error message is provided to the user.
3. **Natural Language Processing:**
   - The text query is processed by spaCy to extract the start and end locations. The system matches the recognized entities (locations) to those in the graph.
4. **Pathfinding:**
   - The graph is used to determine the possible paths from the start location to the end location. The system explores all paths and outputs the results in a user-friendly format.
5. **Text-to-Speech Output:**
   - The system uses pyttsx3 to speak the navigation directions aloud, guiding the user through the campus.

Speed and Text Procession Flowchart:



Working Mechanism:

Following is the code, written to build this project. Converting the Project idea into code.

C:\Users\HP8\Desktop\NavigationSystem.py

NavigationSystem.py

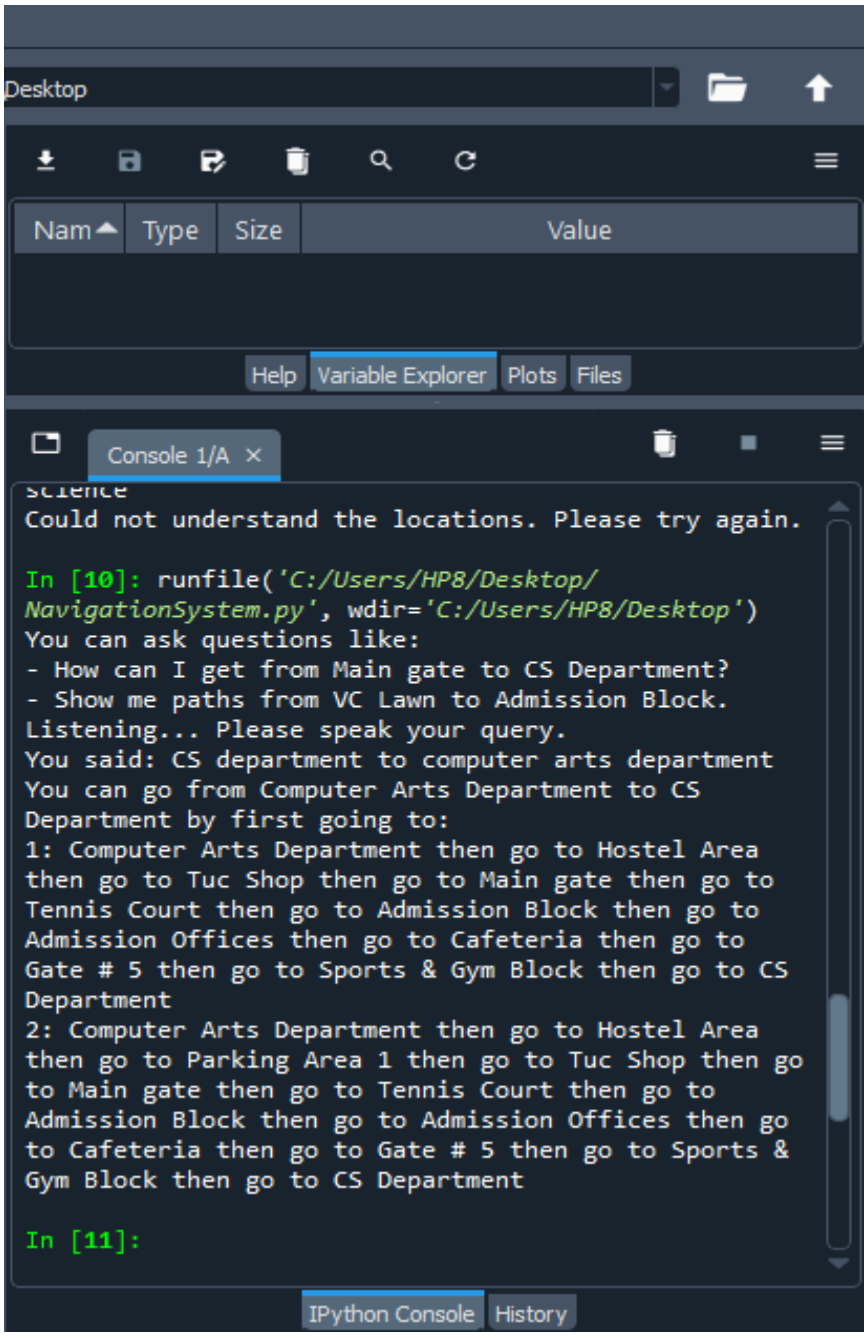
```python
import spacy
import speech_recognition as sr
import pyttsx3

def text_to_speech(text):
    """
    Convert text to speech.
    """
    engine = pyttsx3.init()
    engine.say(text)
    engine.runAndWait()

def speech_to_text():
    """
    Convert speech to text using the microphone.
    """
    recognizer = sr.Recognizer()
    with sr.Microphone() as source:
        print("Listening... Please speak your query.")
        text_to_speech("Listening... Please speak your query.")
        try:
            audio = recognizer.listen(source, timeout=5)
            query = recognizer.recognize_google(audio)
            print(f"You said: {query}")
            return query
        except sr.UnknownValueError:
            text_to_speech("Sorry, I did not understand that.")
            return None
        except sr.RequestError:
            text_to_speech("Sorry, there was an issue with the speech recognition service.")
            return None
        except sr.WaitTimeoutError:
            text_to_speech("No input detected. Please try again.")
            return None

            return None

def extract_locations(query, node_names):
    """
    Extract starting and ending locations from the user's natural language query.
    """
    nlp = spacy.load("en_core_web_sm")
    extracted_locations = []

    # Match node names in the query
    for name in node_names.values():
        if name.lower() in query.lower():
            extracted_locations.append(name)

    # Extract start and end locations
    if len(extracted_locations) >= 2:
        start, end = extracted_locations[:2]  # First two locations
        start_key = next(key for key, value in node_names.items() if value.lower() == start.lower())
        end_key = next(key for key, value in node_names.items() if value.lower() == end.lower())
        return start_key, end_key
    else:
        return None, None

def find_all_paths(graph, start, end, path=[]):
    """
    Find all paths from the start node to the end node in a graph.
    """
    path = path + [start]
    if start == end:
        return [path]
    if start not in graph:
        return []
    paths = []
    for node in graph[start]:
        if node not in path:
```

```
        text_to_speech(f"Path {i}: {readable_path}")

def main():
    graph = {
        "A": {"B": 2, "C": 3},
        "B": {"A": 2, "E": 1},
        "C": {"A": 3, "M": 5},
        "D": {"E": 6, "W": 3},
        "E": {"B": 1, "D": 6, "F": 2, "G": 2},
        "F": {"E": 2, "H": 3, "I": 5, "J": 4, "K": 6, "L": 2},
        "G": {"E": 2},
        "H": {"F": 3},
        "I": {"F": 5},
        "J": {"F": 4},
        "K": {"F": 6},
        "L": {"F": 2},
        "M": {"C": 5, "N": 2},
        "N": {"M": 2, "O": 3},
        "O": {"N": 3, "P": 2},
        "P": {"O": 2, "Q": 4},
        "Q": {"P": 4, "R": 3, "S": 5, "T": 6},
        "R": {"Q": 3},
        "S": {"Q": 5},
        "T": {"Q": 6, "U": 4},
        "U": {"T": 4, "V": 3},
        "V": {"U": 3},
        "W": {"D": 3, "X": 2},
        "X": {"W": 2}
    }
```

```
        node_names = {
            "A": "Main gate",
            "B": "Tuck Shop",
            "C": "Tennis Court",
            "D": "VC Lawn",
            "E": "Parking Area 1",
            "F": "Hostel Area",
            "G": "Student Affair Office",
            "H": "FM Radio Room",
            "I": "Law Department",
            "J": "Computer Arts Department",
            "K": "Susan B Reading Room",
            "L": "Educational Department",
            "M": "Admission Block",
            "N": "Admission Offices",
            "O": "Cafeteria",
            "P": "Gate # 5",
            "Q": "Sports & Gym Block",
            "R": "Science Block",
            "S": "CS Department",
            "T": "IR Department",
            "U": "Masjid",
```

```
        query = speech_to_text()
        if not query:
            return

        start, end = extract_locations(query, node_names)

        if not start or not end:
            message = "Could not understand the locations. Please try again."
            print(message)
            text_to_speech(message)
            return

        print_all_paths(start, end, graph, node_names)


    if __name__ == "__main__":
        main()
```

Output:


Desktop
Nam
Type
Size
Value
Help
Variable Explorer
Plots
Files
Console 1/A
Could not understand the locations. Please try again.
In [10]: runfile('C:/Users/HP8/Desktop/NavigationSystem.py', wdir='C:/Users/HP8/Desktop')
You can ask questions like:
- How can I get from Main gate to CS Department?
- Show me paths from VC Lawn to Admission Block.
Listening... Please speak your query.
You said: CS department to computer arts department
You can go from Computer Arts Department to CS Department by first going to:
1: Computer Arts Department then go to Hostel Area then go to Tuc Shop then go to Main gate then go to Tennis Court then go to Admission Block then go to Admission Offices then go to Cafeteria then go to Gate # 5 then go to Sports & Gym Block then go to CS Department
2: Computer Arts Department then go to Hostel Area then go to Parking Area 1 then go to Tuc Shop then go to Main gate then go to Tennis Court then go to Admission Block then go to Admission Offices then go to Cafeteria then go to Gate # 5 then go to Sports & Gym Block then go to CS Department
In [11]:
IPython Console
History

Role of AI in the Campus Navigation System:

Artificial Intelligence (AI) is the backbone of this campus navigation system, enabling it to understand, process, and respond to natural language queries efficiently. The integration of AI makes the system interactive, intuitive, and hands-free, providing a seamless user experience. Here are the key ways in which AI powers the system:

1. Speech Recognition: Turning Speech into Action:

AI enables the system to convert the user's spoken words into text using advanced speech recognition techniques. By leveraging the **SpeechRecognition** library and Google's Web Speech API, the system transcribes voice commands with high accuracy, even in noisy environments. This allows users to interact with the system in a natural, hands-free manner, making the navigation process more accessible and intuitive.

2. Natural Language Processing (NLP): Understanding User Intent

Once the speech is converted into text, AI uses **Natural Language Processing (NLP)** to interpret the meaning behind the user's query. By utilizing the **spaCy** library, the system extracts key details, such as the start and destination locations. NLP allows the system to handle various input phrasing, synonyms, and contextual variations, ensuring accurate comprehension and efficient query processing.

3. Pathfinding Algorithms: Navigating the Campus

AI is essential for determining the best routes within the campus. The system utilizes **Depth-First Search (DFS)** and other AI-based pathfinding algorithms like **Dijkstra's** or **A*** to explore and find the most optimal paths. These algorithms simulate decision-making processes, enabling the system to provide dynamic route suggestions and adapt to changes in the campus layout, such as new paths or blocked routes.

4. Speech Synthesis: Giving Clear, Human-like Directions

AI ensures that the system provides clear and user-friendly spoken directions through **Text-to-Speech (TTS)** synthesis. Powered by **pyttsx3**, the system converts route information into spoken words, allowing users to follow the directions hands-free. The AI-generated speech creates a conversational tone, enhancing user engagement and making the interaction feel more natural.

5. Continuous Learning and Improvement: Evolving with User Interactions

AI can continuously improve over time by learning from user interactions. By collecting

feedback and analyzing query patterns, the system can adapt to new phrases, refine voice recognition, and enhance its understanding of user intent. This form of machine learning will enable the system to become smarter and more accurate with each use, offering better navigation solutions and improving overall user experience.

Challenges and Solutions

1. Speech Recognition Accuracy

Speech recognition can be challenging due to background noise, accents, and unclear pronunciation.

Solution:
The system uses the **SpeechRecognition** library with **timeout** and **error handling** features to manage noise and improve recognition accuracy.

2. Location Extraction

Extracting locations from varied natural language queries is complex, as users may phrase things differently.

Solution:
spaCy NLP processes the query and matches named entities with a predefined list of locations, handling variations in phrasing.

3. Pathfinding Complexity

Large graphs with many nodes and edges increase the complexity of finding paths between locations.

Solution:
The system uses **Depth-First Search (DFS)** for smaller graphs, and advanced algorithms like **Dijkstra's** or **A*** can be implemented for optimization in larger datasets.

Future Improvements

1. Optimization of Pathfinding

Goal: Enhance pathfinding by implementing **Dijkstra's** or **A*** algorithms to find the shortest and most efficient routes.

2. Voice Command Enhancements

Goal: Expand voice command functionality, allowing users to ask for specific tasks like **“Show me the nearest restroom”** or **“Find the fastest route.”**

3. User Interface

Goal: Develop a **GUI** or **mobile app** to provide users with a seamless interaction through both **voice** and **visual** interfaces.

4. Multi-Language Support

Goal: Enable support for multiple **languages** or **dialects** to accommodate a broader audience and make the system more inclusive.

Conclusion:

This AI-based navigation system provides an innovative way to assist individuals in navigating complex environments like university campuses.

In this project, the campus navigation system utilizes **graph traversal algorithms** such as DFS to explore and find paths between locations. While the current system is based on a simple **depth-first search** for pathfinding, advanced algorithms like **Dijkstra's** or **A*** could be integrated in the future for more optimized performance. By integrating speech recognition, natural language processing, and text-to-speech synthesis, the system creates an interactive and hands-free experience for users. This project demonstrates the power of AI technologies in solving real-world problems and creating user-friendly applications.