AI and Chatbot for Social Robot ”Noor”

Samaa Khair, Mohamed Khaled

June 8, 2024

1 Introduction

This report details the proposed AI and chatbot functionalities for Nour, the social robot. Our focus will be on computer vision, natural language processing, and the integration of these aspects within the Robot Operating System (ROS).

2 Computer Vision

2.1 AI Models

We propose utilizing pre-trained Convolutional Neural Networks (CNNs) for object recognition and scene understanding. Popular options include YOLOv5 or TensorFlow Object Detection API. These models excel at identifying objects within an image or video frame.

2.2 Training Data

The chosen CNN model will require training data specific to Nour’s environment. This data will consist of labeled images or videos containing objects Nour should recognize, like furniture, people, or specific landmarks.

2.3 Algorithms

During operation, the selected CNN model will receive live video feed from the RealSense camera. The model will then identify objects within the frame and provide their corresponding labels. This information will be crucial for tasks like navigation and user interaction.

3 Chatbot

3.1 Language Processing Techniques

Natural Language Processing (NLP) techniques will be employed to enable Nour to understand and respond to user conversations. This involves a combination of techniques like intent recognition, named entity recognition, and sentiment analysis.

3.2 Chatbot Models

We propose exploring pre-trained dialogue models like Rasa or Microsoft Bot Framework. These models can be further customized with Nour-specific responses and conversation flows.

3.3 Tools and Technologies

- **AI Frameworks:** TensorFlow or PyTorch will be used for model training and deployment.
- **ROS Packages:** We will utilize ROS packages like cv_bridge and rospy to bridge the gap between computer vision tasks and ROS communication.
- **Custom Software:** Custom software modules might be needed to handle specific functionalities like intent classification or dialogue management within the chatbot.

4 ROS Nodes and Scripts

4.1 Addressed Functions

The following functions are addressed in the provided ROS nodes and scripts:

- **Initialization and Logging:** The minimal node initializes the ROS environment and logs its status.
- **Image Processing:** The object detection node processes images from a camera, detects objects using a pre-trained model, and publishes the detected objects.
- **Speech Transcription:** The speech transcription node converts audio input into text using a speech recognition model.
- **Chatbot Interaction:** The chatbot node analyzes the transcribed speech, determines the appropriate response using sentiment analysis and a pre-trained language model, and publishes the response.

4.2 Packages Used

The scripts make use of the following packages:

- `rospy`: ROS client library for Python.
- `sensor_msgs`: ROS message types for sensor data.

- `std_msgs`: Standard ROS message types.
- `cv_bridge`: Bridge between ROS and OpenCV.
- `cv2`: OpenCV library for image processing.
- `torch` and `torchvision`: PyTorch library for deep learning models and utilities.
- `whisper`: Whisper library for speech recognition.
- `transformers`: Hugging Face Transformers library for NLP models.
- `openai`: OpenAI API for accessing GPT models.

4.3 Interfacing with the Robot Functions

The ROS nodes are interfaced with the rest of the robot functions as follows:

- **Camera Node:** Captures live video feed and publishes it to the `/usb_cam/image_raw` topic.
- **Object Detection Node:** Subscribes to the `/usb_cam/image_raw` topic to receive images, processes the images to detect objects, and publishes the detected objects to the `/detected_objects` topic.
- **Speech Transcription Node:** Converts audio input to text and publishes the transcribed speech to the `/transcribed_speech` topic.
- **Chatbot Node:** Subscribes to the `/transcribed_speech` topic to receive transcribed speech, generates a response, and publishes the response to the `/prompt` topic.

4.4 System Integration

The integration of the described functionalities within the ROS framework ensures that:

- The camera node provides real-time image data to the object detection node.
- The object detection node processes the images and outputs recognized objects.
- The speech transcription node converts audio input to text, which is then processed by the chatbot node.
- The chatbot node generates appropriate responses based on the transcribed speech and determined sentiment, allowing Nour to interact with users.

4.5 Data Flow and Integration

- **Initial Setup**
  - Create or update existing launch files to include the new nodes. This ensures all components start correctly when the system is launched.
- **Data Flow**
  - **Camera Feed to Object Detection:**
    * The camera node captures real-time video feed and publishes it to a ROS topic, e.g., `/usb_cam/image_raw`.
    * The object detection node subscribes to this topic, processes the images to detect objects, and publishes the detected objects to the `/detected_objects` topic.
  - **Audio Input to Text Transcription:**
    * The speech transcription node receives audio input, transcribes it to text using the Whisper model, and publishes the transcribed text to the `/transcribed_speech` topic.
  - **Text to Chatbot Response:**
    * The chatbot node subscribes to the `/transcribed_speech` topic to receive the transcribed text, analyzes the text using sentiment analysis and a GPT model, generates an appropriate response, and publishes the response to the `/prompt` topic.

5 Code Quality and Documentation

5.1 Clear Code with Comments

All code provided for the AI and chatbot functionalities includes clear comments to explain the purpose and functionality of each section. This ensures that other developers can easily understand and maintain the code.

5.2 Used Packages

The provided scripts utilize several packages to achieve the desired functionalities. These packages are essential for handling tasks such as image processing, speech transcription, and natural language processing. The main packages used include `rospy`, `sensor_msgs`, `cv_bridge`, `torch`, `whisper`, `transformers`, and `openai`.

6 Topics, Nodes, and Services Shared with the Team

6.1 Detected Object Topic

The `/detected_objects` topic is shared with the control team. This topic publishes information about the objects detected by the object detection node. The control team uses this information to make decisions regarding the robot's movements and interactions with its environment.

6.2 Personality Topic

The `/personality` topic is shared with the mechanical team. This topic publishes the determined personality of the robot based on the sentiment analysis of the transcribed speech. The mechanical team uses this information to display the appropriate personality on the robot's screen, enhancing the user experience by showing a friendly, empathetic, or professional demeanor.

7 Topics, Nodes, and Services Needed from Other Teams

To ensure the integrated functionalities work seamlessly, the following topics, nodes, and services are required from the work of other teams:

7.1 Navigation Topic

We need access to the `/navigation_commands` topic managed by the navigation team. This topic will provide commands necessary for the robot to move to specific locations based on the detected objects and user interactions.

7.2 Battery Status Service

A service that provides the current battery status of the robot is crucial. This will help manage and optimize the robot's activities, ensuring it doesn't run out of power during critical operations.

7.3 User Interaction Node

We require a node that handles user interaction data, such as touch inputs or button presses, to complement the voice interaction managed by the chatbot. This will provide a more comprehensive user interaction experience.

7.4 Environment Mapping Topic

An environment mapping topic is needed to share data about the robot's surroundings. This will assist the object detection and navigation functionalities by providing context about the robot's location and environment.

8 Code Quality and Documentation

8.1 Clear Code with Comments

All code provided for the AI and chatbot functionalities includes clear comments to explain the purpose and functionality of each section. This ensures that other developers can easily understand and maintain the code.

8.2 Used Packages

The provided scripts utilize several packages to achieve the desired functionalities. These packages are essential for handling tasks such as image processing, speech transcription, and natural language processing. The main packages used include `rospy`, `sensor_msgs`, `cv_bridge`, `torch`, `whisper`, `transformers`, and `openai`.

9 Functional Testing and Demonstration

The developed functionalities have been thoroughly tested to ensure they work as intended. Demonstrations of object detection, speech transcription, and chatbot interaction were performed to validate the integration and functionality of each component. These demonstrations showed that the robot could detect objects in real-time, transcribe spoken language accurately, and generate appropriate responses to user interactions.

10 Conclusion

The integration of advanced computer vision and natural language processing capabilities into the NOOR_RP repository has significantly enhanced the functionality of the social robot Nour. By leveraging pre-trained models and robust ROS integration, Nour can now interact with its environment and users in a more intelligent and natural manner.