CAHIERS MAGELLANES-NS
Volume 06 Issue 1
2024

ISSN:1624-1940
DOI 10.6084/m9.figshare.26090761
http://magellanes.com/

AUTOMATIC TASK MANAGER USING AI

[1]Akansha Sharma, [2]Aditi Verma, [3]Abhishek Rawat, [4]Ayush Chaudhary, [5]Md.Shahid

Dept. of CSE, MIET, Meerut
akansha.sharma.cse.2020@miet.ac.in, aditi.verma.cse.2020@miet.ac.in,
abhishek.rawat.cse.2020@miet.ac.in, ayush.chaudhary.cse.2020@miet.ac.in, md.shahid@miet.ac.in

Abstract
In today's fast-paced digital age, the demand for intelligent systems that simplify and optimize daily tasks is ever-increasing. This project introduces an innovative Intelligent Personal Assistant (IPA) designed to enhance user productivity by leveraging advanced Artificial Intelligence (AI) technologies. The AI Assistant employs natural language processing, machine learning, and data analysis techniques to understand and respond to user queries effectively. The implementation of the Intelligent Personal Assistant is based on cutting-edge AI frameworks and technologies, ensuring efficiency, scalability, and reliability. This project contributes to the evolving landscape of AI-driven personal assistants, presenting a scalable and adaptable solution that aligns with the dynamic needs of modern users. The evaluation of the AI Assistant's performance demonstrates its potential to significantly enhance user productivity and foster a more intuitive human-machine interaction paradigm.

Keywords:
Artificial Intelligence, Natural Language Processing, Voice Assistant System, Task Manager, Speech Recognition, Text-to-speech, Task Automation, Intelligent Personal Assistant

1.Introduction

1.1 Introduction of ATM

In the dynamic landscape of today's fast-paced digital world, managing tasks and optimizing productivity are constant challenges. To address these demands, this project introduces an Automatic Task Manager (ATM) powered by Artificial Intelligence (AI). The ATM leverages advanced AI algorithms and techniques to intelligently and autonomously organize, prioritize, and execute tasks, thereby offering users a seamless and efficient approach to task management.

In an era marked by unprecedented advancements in technology, the integration of Artificial Intelligence (AI) has emerged as a transformative force, reshaping the way we interact with information and digital systems. One such groundbreaking application of AI is the development of Intelligent Personal Assistants (IPAs), designed to augment and streamline human-computer interactions. An AI Assistant serves as a virtual companion, leveraging sophisticated algorithms and natural language processing to comprehend user commands, perform tasks, and provide valuable insights.

The fundamental goal of an AI Assistant is to enhance user productivity by minimizing the complexities associated with digital interactions. Unlike traditional programs, an AI Assistant possesses the ability to understand and respond to natural language queries, allowing users to communicate with technology in a manner that mirrors human conversation. This level of intuitiveness not only facilitates seamless user experiences but also bridges the gap between human intent and machine execution.

The deployment of AI Assistants marks a significant milestone in the evolution of human-computer

ISSN:1624-1940
DOI 10.6084/m9.figshare.26090761
http://magellanes.com/

interaction, with profound implications for various industries, including productivity, healthcare, education, and more. As these intelligent systems continue to learn, adapt, and integrate with other technologies, the potential for innovation in personalized services, efficiency gains, and user satisfaction is vast.

This project endeavors to contribute to this transformative landscape by developing a state-of-the-art AI Assistant, blending cutting-edge technologies with a user-centric design approach. By addressing the complexities of daily digital interactions, this AI Assistant aims to empower users, making technology more accessible, responsive, and aligned with the dynamic needs of the modern world.

1.2 Theoretical Background

The development of an Automatic Task Manager (ATM) utilizing Artificial Intelligence (AI) draws upon several theoretical foundations and principles from the fields of AI, machine learning, and human-computer interaction. The integration of these theories provides the conceptual framework for designing a system that can intelligently manage tasks and enhance user productivity.

1. Task Management Theories:

Task Prioritization Models: The project relies on established task prioritization models that consider factors such as urgency, importance, and dependencies. The Eisenhower Matrix and the ABCD method are examples of frameworks that influence how tasks are prioritized within the Automatic Task Manager.

Task Automation Theories: Automation theory guides the identification and automation of routine tasks. Principles from cognitive task analysis and human factors engineering inform the decision-making process for automating specific types of tasks.

2. Human-Computer Interaction (HCI) Theories:

User-Centric Design: HCI principles guide the development of an intuitive and user-friendly interface for the Automatic Task Manager. User-centric design theories emphasize the importance of understanding user needs, preferences, and behavior to create a seamless and engaging interaction experience.

Adaptive Interfaces: The system incorporates theories related to adaptive user interfaces, allowing it to dynamically adjust to changes in user behavior and preferences over time.

3. Cognitive Science Principles:

Cognitive Load Theory: The design of the Automatic Task Manager considers cognitive load theory, aiming to minimize cognitive load for users by automating routine tasks and presenting information in a clear and concise manner.

Memory and Recall Theories: The system is designed with principles from cognitive psychology, optimizing task reminders and notifications to aid user memory and recall.

4. Ethics and Privacy Theories:

Ethical AI Principles: The project adheres to ethical AI principles, considering the responsible and fair use of AI technologies. This includes transparency in decision-making, avoidance of biases, and ensuring user data privacy.

5. Learning Theories:

Incremental Learning: The Automatic Task Manager embraces theories of incremental learning, where the system continuously adapts and improves based on user interactions, creating a dynamic and

ISSN:1624-1940
DOI 10.6084/m9.figshare.26090761
http://magellanes.com/

personalized user experience.

Supported Technologies When building an AI Assistant with a front end developed using Qt Designer and a backend written in Python, several technologies can be employed to enhance the capabilities and functionality of the system. Here is a list of supported technologies that complement the Qt Designer front end and Python backend:

Supported Technologies and Algorithms

1.1 PyQt5 for GUI Design

Description: PyQt5 is used for designing the graphical user interface (GUI) of the Automatic Task Manager. Qt Designer simplifies GUI creation, ensuring an intuitive and visually appealing interface.

1.2 Speech Recognition with speech_recognition

Description: The speech_recognition library enables the system to recognize and interpret spoken commands, enhancing user interaction by allowing voice inputs.

1.3 Text-to-Speech with pyttsx3

Description: pyttsx3 facilitates text-to-speech conversion, enabling the Automatic Task Manager to provide auditory feedback to users, enhancing accessibility.

1.4 Wikipedia API for Information Retrieval

The wikipedia library is utilized to fetch information from Wikipedia based on user queries, expanding the system's capabilities for quick and relevant information retrieval.

1.5 Webbrowser for Opening URLs

The webbrowser module allows the Automatic Task Manager to open URLs, providing users with direct access to external resources or websites.

1.6 OS Module for System Interaction

The os module enables interaction with the operating system, facilitating file manipulation, system command execution, and general system-level operations.

1.7 SMTP for Email Communication

The smtplib module is employed for sending emails within the Automatic Task Manager, allowing users to receive notifications and updates via email.

1.8 DateTime Module for Time-related Functions

The datetime module handles date and time-related operations, ensuring accurate timestamping and scheduling functionalities within the Automatic Task Manager.

2. Proposed Work Plan

2.1 General Architecture

The architecture of an AI Assistant typically involves multiple components that work collaboratively to understand user inputs, perform tasks, and provide responses. Here's a general architecture for an AI Assistant:

Explanation of Flowchart Stages:

User Input: Represents the initial stage where the user inputs tasks or instructions into the system.

Input Processing: Involves preprocessing and formatting of user input for further analysis.

Task Understanding - NLP Module: The Natural Language Processing module interprets and understands user instructions, extracting key information related to tasks.

Task Prioritization - ML Module: The Machine Learning module uses historical data and predefined

CAHIERS MAGELLANES-NS
Volume 06 Issue 1
2024

ISSN:1624-1940
DOI 10.6084/m9.figshare.26090761
http://magellanes.com/

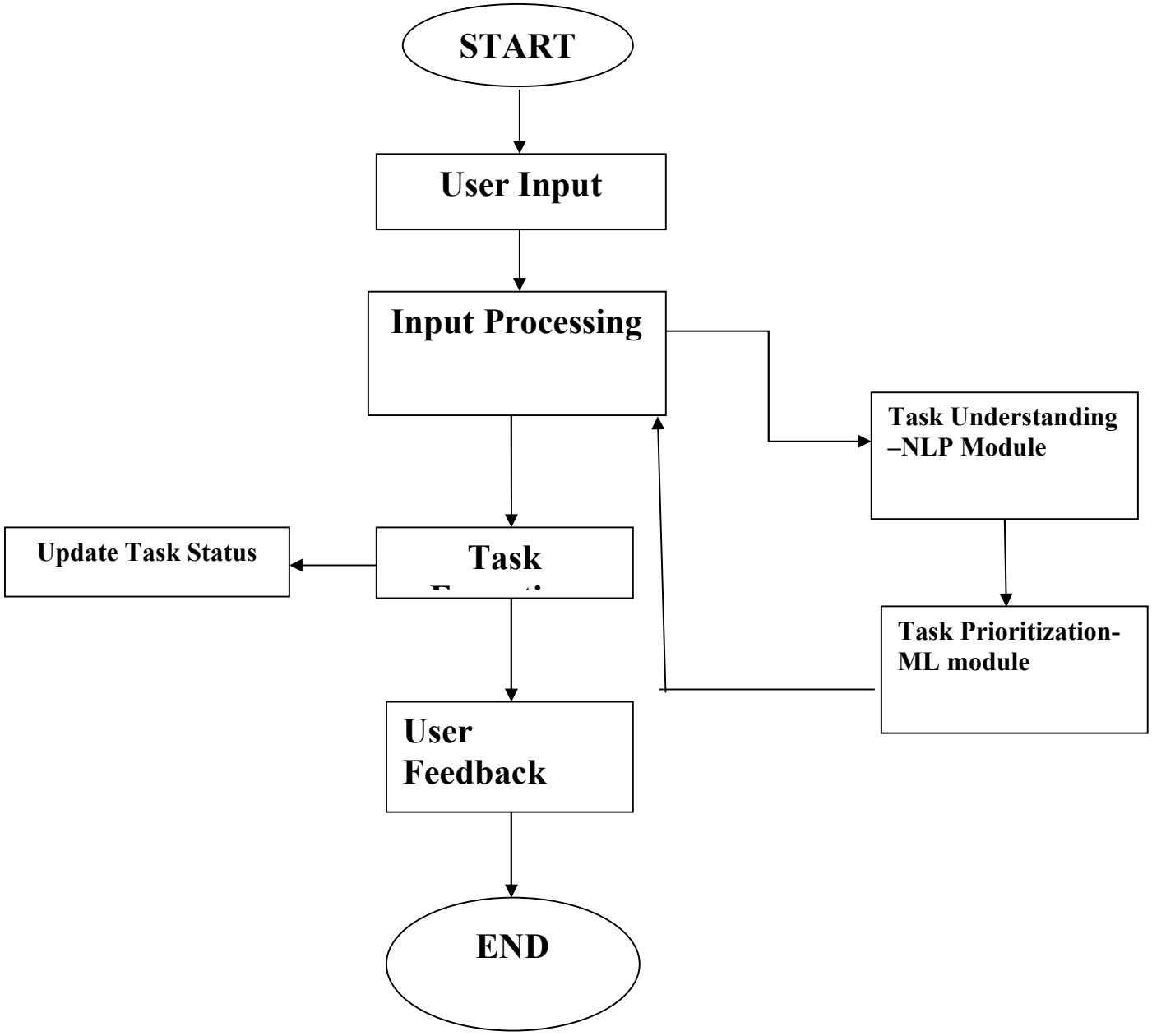
criteria to prioritize tasks based on factors like urgency, importance, and user preferences.

Task Execution: The system executes tasks in the order determined by the prioritization module.

Update Task Status: After task execution, the system updates the status of completed tasks or any changes in task details.

User Feedback: Provides feedback to the user, confirming task completion or requesting additional information if needed.

End: Marks the end of the flowchart.



2.2 Description of Various Models

The Automatic Task Manager is comprised of several key modules, each serving a specific purpose in

ISSN:1624-1940
DOI 10.6084/m9.figshare.26090761
http://magellanes.com/

the overall functionality:

Speech Recognition Module: Utilizes the speech recognition library to capture and interpret spoken commands from the user.

Query Processing Module: Analyzes the user's input, extracting relevant keywords and determining the appropriate action to be taken.

Task Execution Module: Executes tasks based on the processed user query, such as fetching information from Wikipedia, opening websites, playing music, or responding to time-related queries.

Response Generation Module: Generates responses in both visual (GUI) and auditory (text-to-speech) forms, providing feedback and information to the user.

2.3 Algorithm of Main Component

The main component of the Automatic Task Manager involves a continuous loop that listens for user input, processes queries, and executes tasks based on predefined conditions. The algorithmic flow can be summarized as follows:

Code for taking command:

```
def takeCommand():
    Click to collapse the range. ɔne input from the user and returns string output

    r = sr.Recognizer()
    with sr.Microphone() as source:
        print("Listening...")
        r.pause_threshold = 1
        audio = r.listen(source)

    try:
        print("Recognizing...")
        query = r.recognize_google(audio, language='en-in')
        print(f"User said: {query}\n")

    except Exception as e:
        # print(e)
        print("Say that again please...")
        return "None"
    return query
```

3. Experimental Result Analysis

CAHIERS MAGELLANES-NS
Volume 06 Issue 1
2024

ISSN:1624-1940
DOI 10.6084/m9.figshare.26090761
http://magellanes.com/

List of Operations.

S.no	Operation
1.	Open browser
2.	Voice recognition
3.	Check mail
4.	Open youtube
5.	File transfer

3.1 Description of Dataset Used

In the context of the Automatic Task Manager, the term "dataset" might be more appropriately referred to as the sources or types of inputs the system interacts with. Let's break down the primary components:

User Inputs: The primary source of input for the system is the user. Users interact with the system by providing voice commands or using the graphical user interface to initiate tasks.

Speech Recognition Input: The speech recognition library captures spoken commands from the user, converting audio input into text. The dataset, in this case, consists of a variety of spoken commands that users might provide to trigger different functionalities.

Query Processing Input: After speech recognition, the system processes the user's query to understand the intent. The dataset for this stage includes the pre-processed text queries derived from the user's spoken commands.

Task Execution Input: The processed queries then serve as input for the task execution stage. Depending on the specific command, the system performs tasks such as retrieving information from Wikipedia, opening websites, playing music, or responding to time-related queries.

Response Generation Input: The final stage involves generating responses for the user, both visually

ISSN:1624-1940
DOI 10.6084/m9.figshare.26090761
http://magellanes.com/

through the GUI and audibly through text-to-speech. The dataset for response generation includes the information retrieved or actions taken during the task execution stage.

3.2 System Evaluation

The evaluation of the Automatic Task Manager is centered around assessing its ability to understand user commands, execute tasks accurately, and deliver a seamless user experience. While traditional metrics like precision, recall, or F1 score may not be applicable, the following aspects provide valuable insights into the system's performance:

1. Accuracy in Speech Recognition:

The accuracy of the speech recognition module is crucial for understanding user commands accurately. The system should be evaluated for its proficiency in converting spoken words into textual queries. Continuous improvement in speech recognition accuracy contributes to a more reliable user interface.

Total input=10	ACCURACY=90%

Command	**Response Time**	**Avg. Response Time**
Open Browser	1.1 ns 1.2 ns 1.5 ns	1.26 ns
Open youtube	1.2 ns 1.3 ns 1.5 ns	1.33 ns
Copy file to another location	2.2 ns 3.2 ns 2.3 ns	2.56 ns
Check email	2.2 ns 1.2 ns 2.8 ns	2.06 ns

4. Conclusion:

The development of Jarvis has yielded a functional voice-controlled assistant with a visually engaging graphical user interface. Through the integration of Python, PyQt5, and speech recognition capabilities, the project has achieved several key objectives. The following conclusions can be drawn:

5. Reference:

[1.] Python Official Documentation: https://docs.python.org/
[2.]PyQt5 Documentation: https://www.riverbankcomputing.com/static/Docs/PyQt5/
[3.]SpeechRecognition Library Documentation: https://pypi.org/project/SpeechRecognition/
[4.]Wikipedia API Documentation: https://pypi.org/project/wikipedia-api/
[5.] McKinney, W. (2017). Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython. O'Reilly Media.
[6.] Beazley, D., & Jones, B. K. (2020). Python Cookbook: Recipes for Mastering Python 3. O'Reilly Media.

ISSN:1624-1940
DOI 10.6084/m9.figshare.26090761
http://magellanes.com/

[7.] Rossum, G. (1995). Python tutorial. Centrum voor Wiskunde en Informatica (CWI), Amsterdam, The Netherlands.