APPLICATION OF LARGE LANGUAGE MODELS AND NATURAL LANGUAGE UNDERSTANDING IN MULTI-AGENT TOURIST GUIDE FOR GABROVO

Iliya Iliev Nedelchev

University of Plovdiv "Paisii Hilendarski"

Abstract

This paper explores the integration of Large Language Models (LLM) and Natural Language Understanding (NLU) within a multi-agent tourist guide for Gabrovo. In the context of the digital transformation of the tourism industry, this study examines how advanced NLP technologies enhance the tourist experience.

The multi-agent tourist guide, powered by LLM and NLU, delivers personalized recommendations, historical insights, and real-time assistance to travelers. This research emphasizes the transformative potential of these technologies in tourism by fostering engagement, promoting cultural immersion, and streamlining navigation.

Furthermore, the study highlights the broader implications of AI-driven language models and understanding in preserving cultural heritage. By harnessing LLM and NLU, this tourist guide redefines exploration in Gabrovo and symbolizes a shift in how individuals connect with cultural legacies worldwide.

In essence, this paper underscores the significant impact of technology in reshaping multi-agent tourist guides, enriching traveler experiences, and preserving cultural identities in destinations such as Gabrovo.

Keywords: natural language understanding, large language models, Rasa, OpenAI ChatGPT.

INTRODUCTION

The tourism industry stands on the precipice of a new era, characterized by the convergence of advanced technology and the enduring spirit of exploration. In an era where travelers seek increasingly immersive and personalized experiences, technology plays an integral role in reshaping the manner in which we discover and engage with destinations. Within this context, we embark on an exploration of the strategic integration of Large Language Models (LLM) and Natural Language Understanding (NLU) technologies, combined with a robust .NET 6 REST API, to create a multi-agent tourist guide. This guide is set against the enchanting backdrop of Gabrovo and its encompassing region.

Gabrovo, with its illustrious tapestry of cultural heritage and natural splendor, provides an ideal canvas for our comprehensive examination. Here, the nexus of tradition and innovation is palpable, mirroring the very essence of our inquiry. Our objective is to unveil the transformative potential of LLM and NLU technologies, orchestrated through a .NET 6 REST API, within the tourism sector. Our primary focus is on elevating the visitor experience to uncharted levels of sophistication.

As an integral component of our case study, we will harness three pioneering technologies: Rasa, Jason, and ChatGPT. Rasa, renowned for its precision in Natural Language Understanding (NLU) and dialogue management, serves as the bedrock of our multi-agent tourist guide, ensuring unparalleled intent recognition and the seamless flow of conversation. Jason, an intelligent agent platform, communicates efficiently with a .NET 6 REST API, thereby enhancing the guide's adaptability and responsiveness to diverse traveler needs. This REST API, in turn, acts as the linchpin of our architecture, facilitating communication between Jason and the formidable ChatGPT, a Large Language Model (LLM). ChatGPT enriches our guide with informative and engaging responses, culminating in a seamless and immersive experience for the discerning traveler.

The multi-agent tourist guide, fortified by the capabilities of Rasa, Jason, ChatGPT, and the orchestrating .NET 6 REST API, stands as a testament to the symbiotic relationship between technology and tourism. It not only disseminates information but fosters engagement, providing not just directions but cultural immersion. Throughout the course of this paper, we will elucidate how these technologies empower travelers with personalized recommendations, historical narratives, and real-time assistance, all finely tuned to their unique preferences and requirements.

Beyond the immediate impact on the tourism landscape, our exploration extends to the broader implications of AI-driven language models and understanding, particularly in the context of cultural heritage preservation and celebration. Our findings shed light on how, through the adept utilization of Rasa, Jason, ChatGPT, and .NET 6 REST API, the multi-agent tourist guide redefines the very essence of exploration in Gabrovo, while simultaneously establishing a precedent for how individuals worldwide engage with and cherish their cultural legacies.

In an epoch defined by technology-driven transformation, our study of the multi-agent tourist guide transcends the realm of scholarly inquiry, serving as a testament to the evolving dynamics between travelers, technology, and the timeless pursuit of discovery. As we navigate this intricate landscape, we cordially invite you to accompany us in unraveling the profound impact of these cutting-edge technologies and architecture, ultimately redefining the traveler's experience and reshaping their connection with destinations such as Gabrovo.

EXPOSITION

At the core of our mobile application is a multifunctional tourist guide system designed to offer travelers an all-encompassing experience when exploring Gabrovo. While the application boasts a wide array of functions, including navigation, information, and local resources, the integration of a chatbot and voicebot enhances its utility.

The application serves as a versatile tool, catering to a multitude of traveler needs. It encompasses functionalities such as navigation assistance, event scheduling, local cuisine recommendations, and real-time updates on attractions and events. Users can access a wealth of information about Gabrovo's history, culture, and traditions through an intuitive and user-friendly interface.

As valuable additions to the mobile application, the chatbot and voicebot offer a conversational layer that seamlessly integrates with the application's broader functionalities. These AI-driven agents act as responsive and informative companions, enriching the traveler's experience with tailored interactions.

When users access specific menus within the application, it automatically opens a conversation mode. This mode provides a range of features and information about Gabrovo and its attractions. Travelers can then initiate inquiries, seek recommendations, or engage in casual dialogue to enhance their understanding of the region.

For those who desire a deeper dive into Gabrovo's attractions, the system seamlessly transitions into attraction-specific conversations. Each attraction has its dedicated dialogue space, allowing travelers to explore its history, significance, and practical information.

A standout feature of our system is its proactive and location-based approach. When the user's GPS location is in proximity to a particular attraction, the system dynamically opens a conversation related to that specific attraction. This responsive behavior enhances user engagement by delivering contextually relevant information precisely when it matters most.

The system leverages two advanced conversational AI technologies, Rasa and ChatGPT, in an orchestrated manner. When a user initiates a conversation or query, the system first consults Rasa for intent recognition and dialogue management. If Rasa determines that the user's query cannot be adequately addressed, it gracefully transitions to a fallback strategy, invoking the ChatGPT API.

In essence, our mobile application stands as a comprehensive gateway to Gabrovo, offering a multitude of functions designed to enhance every facet of the traveler's journey. The inclusion of the tourist plan manager ensures an enriched and personalized exploration of Gabrovo, guided by the best of AI-driven technologies while enjoying the broader functionality the application offers.

Rasa Open Source is an open source conversational AI platform that allows you to understand and hold conversations, and connect to messaging channels and third party systems through a set of APIs. It supplies the building blocks for creating virtual (digital) assistants or chatbots [1]. With Rasa, developers can build sophisticated chatbots and virtual assistants that comprehend user inputs, extract intent and entities, and engage users in context-aware conversations.

Key Features:

NLU Engine: Rasa's advanced NLU engine accurately deciphers user queries, enabling effective understanding of intent, entities, and context.

Dialogue Management: It orchestrates dynamic, rule-based, and machine learning-driven conversations, going beyond simple Q&A interactions.

Customization: Rasa's open-source nature allows developers to customize and extend the framework, tailoring solutions to unique requirements.

Thriving Community: A collaborative global community supports Rasa, sharing best practices and contributing to its continuous improvement.

Rasa empowers organizations and developers to create intelligent, context-aware chatbots and virtual assistants across various domains, from customer service to healthcare, fostering innovation and adaptability.

Training Rasa models to deliver intelligent conversations is a pivotal aspect of our multi-agent tourist guide. To achieve this, we employ the standard Rasa REST API for training and inference, but our unique architecture involves a custom-developed Flask REST API to facilitate communication with Rasa agents, each loaded with separate models tailored for distinct purposes.

The Flask framework is actually a glue, a very nice one, that sticks together the amazing Werkzeug and Jinja2 frameworks, responsible for answering requests and presenting the output [2].

Our journey begins with the training of Rasa models, a critical step in ensuring that our chatbot and voicebot are well-equipped to engage users effectively. Rasa's versatile framework allows us to fine-tune models for precise intent recognition, entity extraction, and dialogue management. This extensive training process empowers our agents to comprehend a wide range of user inputs, varying from general queries about Gabrovo to highly specific questions related to individual attractions.

The crux of our architecture lies in the custom Flask REST API, meticulously crafted to orchestrate communication with Rasa agents. This API serves as the central hub through which user requests, inquiries, and interactions are routed to the appropriate Rasa model, ensuring a contextually relevant and accurate response. Additionally, the Flask REST API seamlessly communicates with the .NET 6 REST API, which further enhances the system's capabilities.

Our custom API performs several critical functions:

Model Loading: It manages the loading of distinct Rasa models, each designed to handle specific conversational scenarios. This separation of models optimizes responsiveness and accuracy.

Intent Routing: Based on user inputs, the API intelligently routes requests to the corresponding Rasa agent. For instance, general inquiries find their way to the general conversation model, while attraction-specific questions are directed to the relevant attraction-specific model.

Contextual Conversations: The API maintains context during conversations, enabling seamless transitions between general and attraction-specific dialogues. This ensures that users receive a cohesive and immersive experience as they navigate the application.

To cater to the diverse needs of travelers, we have implemented multiple Rasa models within our custom Flask REST API. These models are specialized for distinct conversational scenarios:

General Conversation Model: This model encompasses conversations of a general nature, offering insights about Gabrovo, its culture, and broader tourist information. It is the go-to model for travelers seeking an overview of the region.

Attraction-Specific Models: In contrast, attraction-specific models are tailored for interactions related to individual attractions in Gabrovo. Each model is dedicated to a particular attraction, allowing travelers to engage in in-depth conversations about the history, significance, and practical details of the specific site.

This separation of models ensures that user queries are channeled to the most relevant Rasa agent, optimizing the accuracy and depth of responses. This multifaceted approach enhances the overall user experience, ensuring that travelers receive precise and engaging information throughout their journey.

In essence, our training and integration strategy empowers our multi-agent tourist guide to provide users with a nuanced and intelligent conversational experience. The combination of Rasa's robust models, coupled with our custom Flask REST API architecture and seamless communication with the .NET 6 REST API, forms the backbone of our system, enabling it to seamlessly address a spectrum of traveler needs, from general inquiries to detailed attraction-specific interactions.

```
from flask import Flask, jsonify, request
import rasa.core.agent
from waitress import serve
def create_app(test_config=None):
    app = Flask(__name__)
    agents = dict()
    endpoints = "endpoints.yml"
@app.route('/initialize/<int:scenario_id>'
, methods=['POST'])
    async def initialize(scenario_id):
        model_path =
request.get_json()['modelPath']
        endpoint_config =
rasa.core.utils.read_endpoints_from_path(e
ndpoints)
        # Intentionally overriding the model
setting in the endpoint config for now
        endpoint_config.model = None
        agents[scenario_id] = await
rasa.core.agent.load_agent(
            model_path = model_path,
            endpoints = endpoint_config
        )
        return '', 204
    @app.route('/send-
message/<string:conversation_id>/<int:scen
ario_id>', methods=['POST'])
    async def
send_message(conversation_id,
scenario_id):
        message =
request.get_json()['message']
        if scenario_id not in agents.keys():
            return '', 403
        results = []
        responses = await
agents[scenario_id].handle_text(message,
None, conversation_id)
        for response in responses:
            if "text" in response:

results.append(response["text"])
        return jsonify(results)
    return app
serve(create_app(), host="rasa.local",
port=8080)
```

This code defines a Flask web application that serves as a communication bridge between your custom application and Rasa chatbot agents. It initializes and manages Rasa agents for different scenarios and allows you to send messages to these agents for generating responses.

Here's a breakdown of the code:

Importing Necessary Libraries:

from flask import Flask, jsonify, request: Imports the Flask framework for building web applications and other essential libraries for handling JSON and HTTP requests.

import rasa.core.agent: Imports the Rasa Core Agent, which is used to load and manage Rasa models.

from waitress import serve: Imports the Waitress web server, which is used to serve the Flask application.

Creating the Flask Application:

create_app(test_config=None): This function creates a Flask application instance. It's a standard pattern in Flask applications to have a function for creating the app instance.

Initializing Agents:

app = Flask(__name__): Initializes the Flask application.

agents = dict(): Initializes a dictionary called agents to store Rasa agents for different scenarios.

endpoints = "endpoints.yml": Specifies the path to an endpoints configuration file, which defines the settings for Rasa agents.

Endpoint for Initializing Rasa Agents:

@app.route('/initialize/<int:scenario_id>', methods=['POST']): Defines a route that listens for HTTP POST requests at the /initialize/<int:scenario_id> URL. This endpoint is used to initialize Rasa agents for specific scenarios.

The initialize(scenario_id) function loads a Rasa agent based on the provided modelPath and stores it in the agents dictionary under the specified scenario_id.

The endpoint_config is read from the endpoints.yml file, and the model setting is intentionally overridden to None. This allows the endpoint to load the agent with a custom model.

The loaded agent is stored in the agents dictionary using the scenario_id as the key.

Endpoint for Sending Messages to Rasa Agents:

@app.route('/send-message/<string:conversation_id>/<int:scenario_id>', methods=['POST']): Defines a route that listens for HTTP POST requests at the /send-message/<string:conversation_id>/<int:scenario_id> URL. This endpoint is used to send messages to Rasa agents for generating responses.

The send_message(conversation_id, scenario_id) function retrieves the incoming message from the request JSON.

It checks if the specified scenario_id exists in the agents dictionary. If not, it returns a 403 status code (Forbidden).

It then uses the loaded Rasa agent to handle the incoming message, specifying the conversation_id. The result is a list of responses.

The responses are extracted, and if a "text" field exists in a response, it is added to the results list.

Finally, the responses are returned as a JSON array.

Running the Flask Application:

serve(create_app(), host="rasa.local", port=8080): This line starts the Waitress web server, serving the Flask application created by create_app(). It listens on host "rasa.local" and port 8080.

This code provides a RESTful API using Flask to initialize and interact with Rasa chatbot agents for different scenarios. It allows you to send messages to these agents and retrieve their responses, making it a crucial component of your multi-agent tourist guide system.

Within our multi-agent tourist guide, we employ SQL Server as a robust storage solution for Rasa models. The table structure adopted for this purpose comprises two essential elements: model titles and the corresponding disk physical paths. This structured approach to model storage enhances efficiency, scalability, and reliability, contributing to an elevated user experience.

ChatGPT, which stands for Chat Generative Pre-trained Transformer, is a large language model–based chatbot developed by OpenAI and launched on November 30, 2022, which enables users to refine and steer a conversation towards a desired length, format, style, level of detail, and language used. Successive prompts and replies, known as prompt engineering, are considered at each conversation stage as a context [3].

We have implemented a seamless interaction with the OpenAI GPT-based chatbot API to enhance the conversational capabilities of our multi-agent tourist guide application.

To initiate this interaction, we utilize the following code:

```
OpenAIAPI api = new OpenAIAPI(setting.OpenAiKey);
var chat = api.Chat.CreateConversation();
chat.AppendUserInput(question);
var result = await chat.GetResponseFromChatbotAsync();
```

The code above can be dissected into the following steps:

Initialization of the OpenAI API:

We begin by initializing an instance of the OpenAIAPI class, where setting.OpenAiKey represents the API key or authentication token required to access the OpenAI GPT model.

Creation of a Conversation:

Within our application, we create a new conversation using the CreateConversation method. This step signifies the initiation of a dialogue with the chatbot, preparing the conversation context.

User Input Appending:

Subsequently, we append the user's input or question (represented by the variable question) to the ongoing conversation. This user input forms the basis for the chatbot's response generation.

Response Retrieval:

Finally, we execute an asynchronous operation, GetResponseFromChatbotAsync(), which sends the user's input to the OpenAI GPT model. We await the response generated by the chatbot, which is based on the user's input and the model's pre-trained knowledge.

In essence, this code represents the pivotal interaction between our tourist guide application and the OpenAI GPT-based chatbot. It enables users to seamlessly seek information, ask questions, and engage in meaningful dialogues about Gabrovo and its attractions. This interaction forms a crucial element of our multi-agent tourist guide, enriching the user experience by providing informative and contextually relevant responses.

CONCLUSION

In conclusion, our paper has presented a comprehensive framework for the development and deployment of a Multi-Agent Tourist Guide tailored to the Gabrovo region and its environs. The primary objective of our endeavor has been to harness the capabilities of Large Language Models (LLMs) and Natural Language Understanding (NLU) to offer a rich and informative experience to travelers.

Our approach has integrated state-of-the-art technologies, with the incorporation of Rasa for intelligent chatbot functionality and the integration of ChatGPT for dynamic and context-aware conversations. The foundation of our system has rested on Jason, a robust agent-based framework, which has played a pivotal role in orchestrating interactions and delivering highly personalized experiences to users.

Notable facets of our system have included the implementation of a .NET 6 REST API to facilitate seamless communication, enabling Jason to engage with Rasa and ChatGPT. Furthermore, we have underscored the critical role of SQL Server in efficiently managing model titles and physical locations, contributing significantly to the responsiveness and dependability of our tourist guide.

Our system has adhered to a proactive approach, initiating conversations based on user GPS location, thereby ensuring that travelers receive timely and contextually relevant information about nearby attractions. Moreover, we have seamlessly integrated a mobile application enriched with voice and chatbot functionalities, enhancing user engagement and accessibility.

Throughout the development process, we have elucidated the practical application of Rasa and ChatGPT, demonstrating their pivotal roles in facilitating conversations, addressing inquiries, and providing insightful details about Gabrovo and its array of attractions.

Our Multi-Agent Tourist Guide signifies a fusion of cutting-edge technologies and pragmatic utility, conceived to offer travelers an intelligent and immersive companion for exploring Gabrovo and its encompassing regions. By harnessing the potency of LLMs, NLU, and agent-based frameworks, we have created a system adept at catering to the diverse requirements of tourists, delivering tailored and information-rich experiences.

REFERENCE

[1] Introduction to Rasa Open Source & Rasa Pro. Available online: https://rasa.com/docs/rasa/ (23 September 2023)

[2] Italo, Maia, Building Web Applications with Flask; Publisher: Packt Publishing Ltd 2015; pp. 1.

[3] ChatGPT. Available online: https://en.wikipedia.org/wiki/ChatGPT (23 September 2023)