Report On

Chatbot for E-commerce Website using Dialogflow

Submitted in partial fulfillment of the requirements of the Course project in Semester VII of Second Year Computer Engineering

by
Varun Satheesh Babu (Roll No. 58)
Siddharth Dhodi (Roll No. 59)
Rohit Walke (Roll No. 70)

Supervisor
Prof. Akshaya Prabhu

University of Mumbai

Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

(2023-24)

Vidyavardhini's College of Engineering & Technology
Department of Computer Engineering

CERTIFICATE

This is to certify that the project entitled “Chatbot for E-commerce Website using Dialogflow” is a bonafide work of "Varun Satheesh Babu (Roll No. 58), Siddharth Dhodi (Roll No. 59), Rohit Walke (Roll No. 70)” submitted to the University of Mumbai in partial fulfillment of the requirement for the Course project in semester VII of Second Year Computer Engineering.

Supervisor

Prof. Akshaya Prabhu

Dr. Megha Trivedi
Head of Department

Abstract

User convenience is one of the most important attributes to the success of a product or service in the industry. Improvement of the end-user experience ranks high among the various concerns of an organization providing a service or a product. This is necessary as the user will only use or subscribe to products or services, they find are not too complicated to use. As a result, companies venture to great lengths to improve their customer's experience using their products or services, in order to establish a strong list of clients and to ultimately lead to success of the organization. Of the many innovations to improve user experience, chatbots are an important part. The application of Natural Language Processing to develop chatbots for websites has become commonplace in today's industry. NLP driven chatbots can help greatly improve user experience by providing help in the cases where the user is stuck while using the website. Chatbots can also perform certain functions at request from the user. Thus, chatbots make websites accessible and user friendly attracting more user traffic towards them.

Contents

1. Introduction

    1.1. Problem Statement

    To create an NLP powered chatbot for an e-commerce website in order to improve user experience.

    1.2. Objective

    This project aims at creating a chatbot that employs NLP for a food ordering website.

    1.3. Scope

    This project may evolve further in the future with added features such as more memory for storing conversations, larger sets of input commands and more human-like conversation skills.

2. Methodology

    2.1. Literature Survey

    ELIZA (1966): ELIZA, developed by Joseph Weizenbaum, is one of the earliest chatbots. It employed simple pattern-matching techniques and acted as a Rogerian psychotherapist. It laid the foundation for subsequent chatbot development.

    ALICE (1995): ALICE, created by Richard Wallace, is a more advanced chatbot that used AIML (Artificial Intelligence Markup Language) for conversation scripting. It became a popular open-source project and inspired the development of many other chatbots.

    Jabberwacky (2005): Jabberwacky is a chatbot that uses a learning approach based on user interactions. It doesn't rely on fixed patterns but learns from conversations to generate responses.

    Cleverbot (2008): Cleverbot, like Jabberwacky, utilizes machine learning techniques to generate responses based on previous conversations. It aims to create more human-like interactions.

    Watson (2010): IBM's Watson is a sophisticated AI system that includes a chatbot component. Watson gained fame by winning the TV game show Jeopardy! and has since been applied to various industries, including healthcare and customer support.

    Siri (2011): Apple's Siri, integrated into its iOS devices, marked a significant step in bringing chatbots to the mainstream. It uses NLP techniques to understand and respond to voice commands.

Google Assistant (2016): Google's Assistant is another mainstream chatbot that uses NLP and machine learning to provide voice and text-based interactions.
GPT-3 (2020): Developed by OpenAI, GPT-3 is a language model that has demonstrated remarkable capabilities in generating human-like text. It's not a chatbot in itself but can be used to power chatbots and virtual assistants.

Rasa (2016): Rasa is an open-source framework for building conversational AI chatbots. It provides tools for NLP, dialogue management, and chatbot development.

Dialogflow (formerly API.ai, 2014): Dialogflow is a Google Cloud service that allows developers to build natural language understanding into their applications. It's widely used for creating chatbots, voice assistants, and more.

Microsoft Bot Framework (2016): Microsoft's Bot Framework is a comprehensive platform for building chatbots and integrating them with various messaging channels.

BERT (2018): BERT (Bidirectional Encoder Representations from Transformers) is a transformer-based model developed by Google. It significantly improved the understanding of context in NLP tasks and has been used in chatbot development.

COVID-19 Chatbots (2020): During the COVID-19 pandemic, several chatbots were developed to provide information and answer questions related to the virus. These chatbots played a crucial role in disseminating information.

2.2. Block Diagram

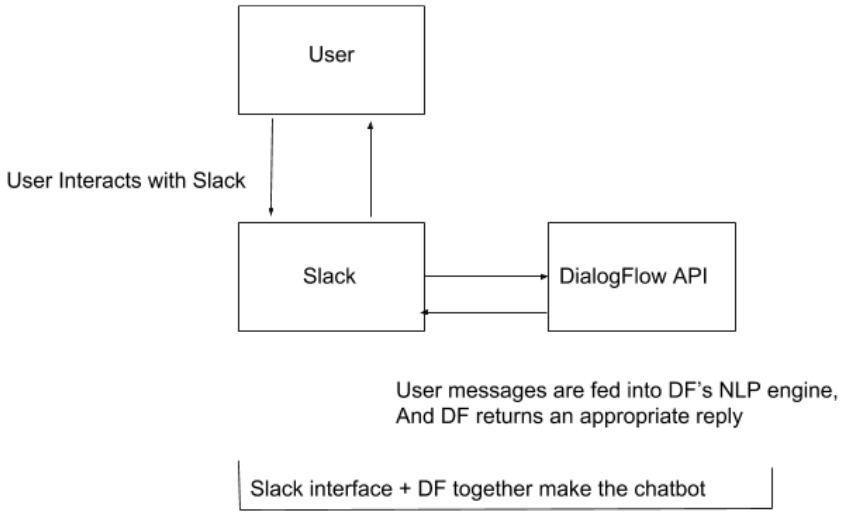

Fig 2.1

2.3. Method of Implementation

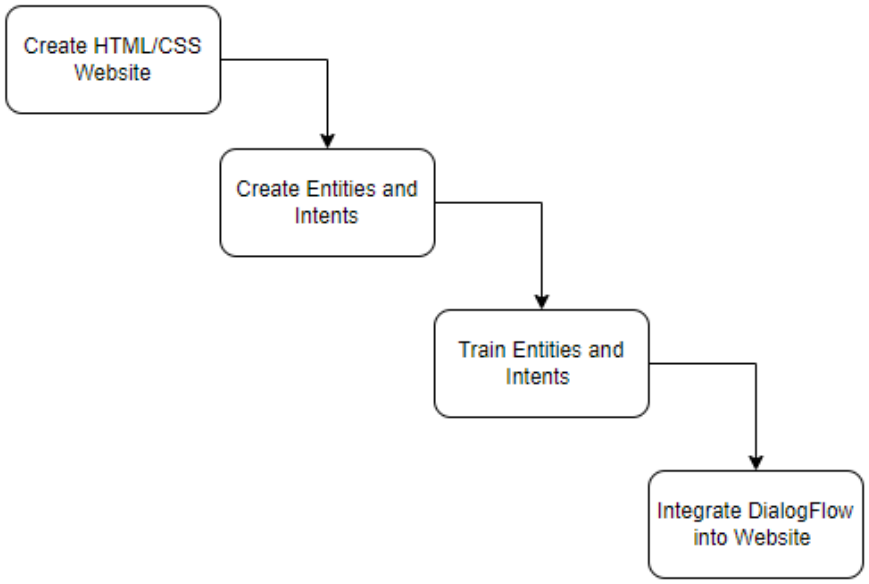

Fig 2.2

The first step in implementing the project is to create a static HTML/CSS website. After which Google's DialogFlow is used to create entities and intents. Thus, created entities and intents are then trained on certain cases. Then the chatbot created is integrated into the HTML/CSS frontend by using FastAPI.

3. Implementation

3.1. Code

```
<html>
<head>
   <link rel="stylesheet" type="text/css" href="styles.css">
   <link
href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;700&display
=swap" rel="stylesheet">
</head>
<body>
        <div class="container">
   <header>
      <nav>
         <a href="#home">HOME</a> |
         <a href="#menu">MENU</a> |
         <a href="#location">LOCATION</a> |
         <a href="#aboutus">ABOUT US</a> |
         <a href="#contactus">CONTACT US</a>
      </nav>
   </header>

   <section id="home">
                  <img src="banner.jpg" >
      <!-- Add images here -->
   </section>
```

```
    <section id="menu">
        <h2>Our Menu</h2>
        <div class="grid-container">
        <img src="menu1.jpg" alt="Menu item 1">
        <img src="menu2.jpg" alt="Menu item 2">
        <img src="menu3.jpg" alt="Menu item 3">
    </div>

    </section>

    <section id="location">
        <h2>Location</h2>
        <p>38 Patli Gali, Mumbai</p>
        <!-- Add map widget here -->
    </section>

    <section id="contactus">
        <h2>Contact Us</h2>
        <p>Got questions? Want to place an order? Call us at +91791234567 or email
us at info@atliqfastfood.com</p>
    </section>


 <iframe class="chat-bot" width="350" height="430" allow="microphone;"
src="https://console.dialogflow.com/api-client/demo/embedded/26e44813-1fb1-
492f-a8b8-6f745839704e"></iframe>
            </div>
</body>
</html>
from fastapi import FastAPI
from fastapi import Request
from fastapi.responses import JSONResponse
import db_helper
```

```
import generic_helper

app = FastAPI()

inprogress_orders = {}

@app.post("/")


async def handle_request(request: Request):

    # Retrieve the JSON data from the request
    payload = await request.json()

    # Extract the necessary information from the payload
    # based on the structure of the WebhookRequest from Dialogflow
    intent = payload['queryResult']['intent']['displayName']
    parameters = payload['queryResult']['parameters']
    output_contexts = payload['queryResult']['outputContexts']
    session_id = generic_helper.extract_session_id(output_contexts[0]["name"])

    intent_handler_dict = {
        'order.add - context: ongoing-order': add_to_order,
        'order.remove - context: ongoing-order': remove_from_order,
        'order.complete - context: ongoing-order': complete_order,
        'track.order - context: ongoing-tracking': track_order
    }

    return intent_handler_dict[intent](parameters, session_id)

def save_to_db(order: dict):
    next_order_id = db_helper.get_next_order_id()
```

```
    # Insert individual items along with quantity in orders table
    for food_item, quantity in order.items():
        rcode = db_helper.insert_order_item(
            food_item,
            quantity,
            next_order_id
        )



        if rcode == -1:
            return -1

    # Now insert order tracking status
    db_helper.insert_order_tracking(next_order_id, "in progress")

    return next_order_id

def complete_order(parameters: dict, session_id: str):
    if session_id not in inprogress_orders:
        fulfillment_text = "I'm having a trouble finding your order. Sorry! Can you place a new order please?"
    else:
        order = inprogress_orders[session_id]
        order_id = save_to_db(order)
        if order_id == -1:
            fulfillment_text = "Sorry, I couldn't process your order due to a backend error. " \
                               "Please place a new order again"
        else:
            order_total = db_helper.get_total_order_price(order_id)
```

```
            fulfillment_text = f"Awesome. We have placed your order. " \
                               f"Here is your order id # {order_id}. " \
                               f"Your order total is {order_total} which you can pay at the time of delivery!"

        del inprogress_orders[session_id]

    return JSONResponse(content={
        "fulfillmentText": fulfillment_text
    })


def add_to_order(parameters: dict, session_id: str):
    food_items = parameters["food-item"]
    quantities = parameters["number"]

    if len(food_items) != len(quantities):
        fulfillment_text = "Sorry I didn't understand. Can you please specify food items and quantities clearly?"
    else:
        new_food_dict = dict(zip(food_items, quantities))

        if session_id in inprogress_orders:
            current_food_dict = inprogress_orders[session_id]
            current_food_dict.update(new_food_dict)
            inprogress_orders[session_id] = current_food_dict
        else:
            inprogress_orders[session_id] = new_food_dict

        order_str = generic_helper.get_str_from_food_dict(inprogress_orders[session_id])
        fulfillment_text = f"So far you have: {order_str}. Do you need anything else?"
```

```
    return JSONResponse(content={
        "fulfillmentText": fulfillment_text
    })


def remove_from_order(parameters: dict, session_id: str):
    if session_id not in inprogress_orders:
        return JSONResponse(content={
            "fulfillmentText": "I'm having a trouble finding your order. Sorry! Can you place a new order please?"
        })


    food_items = parameters["food-item"]
    current_order = inprogress_orders[session_id]

    removed_items = []
    no_such_items = []

    for item in food_items:
        if item not in current_order:
            no_such_items.append(item)
        else:
            removed_items.append(item)
            del current_order[item]

    if len(removed_items) > 0:
        fulfillment_text = f'Removed {",".join(removed_items)} from your order!'

    if len(no_such_items) > 0:
        fulfillment_text = f' Your current order does not have {",".join(no_such_items)}'
```

```
    if len(current_order.keys()) == 0:
        fulfillment_text += " Your order is empty!"
    else:
        order_str = generic_helper.get_str_from_food_dict(current_order)
        fulfillment_text += f" Here is what is left in your order: {order_str}"

    return JSONResponse(content={
        "fulfillmentText": fulfillment_text
    })
```

```
def track_order(parameters: dict, session_id: str):
    order_id = int(parameters['order_id'])
    order_status = db_helper.get_order_status(order_id)
    if order_status:
        fulfillment_text = f"The order status for order id: {order_id} is: {order_status}"
    else:
        fulfillment_text = f"No order found with order id: {order_id}"

    return JSONResponse(content={
        "fulfillmentText": fulfillment_text
    })
```

3.2. Output

Fig 3.1

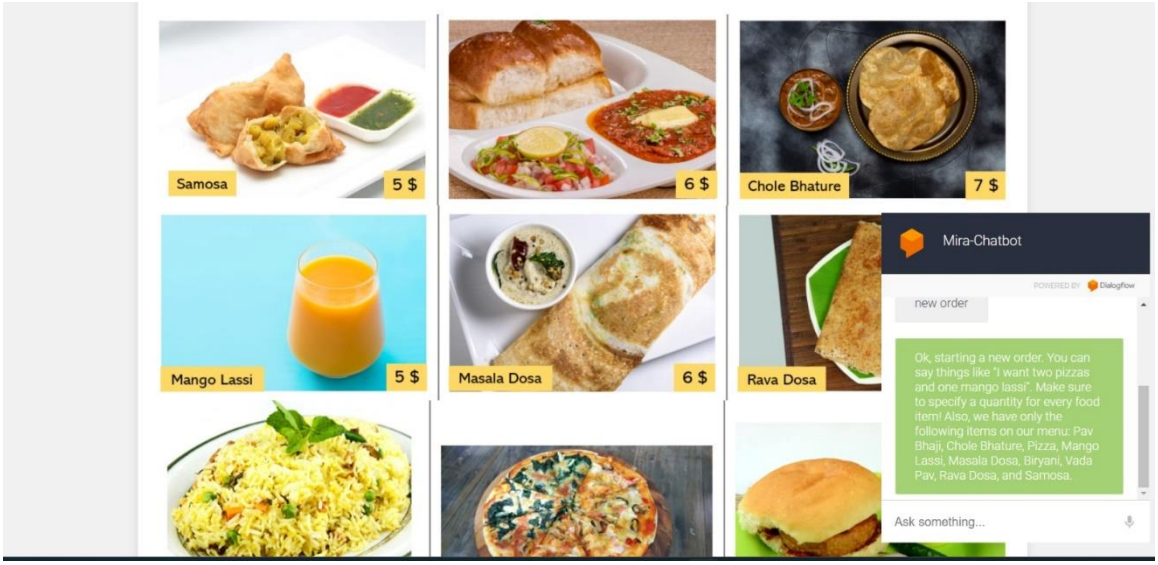

Fig 3.2

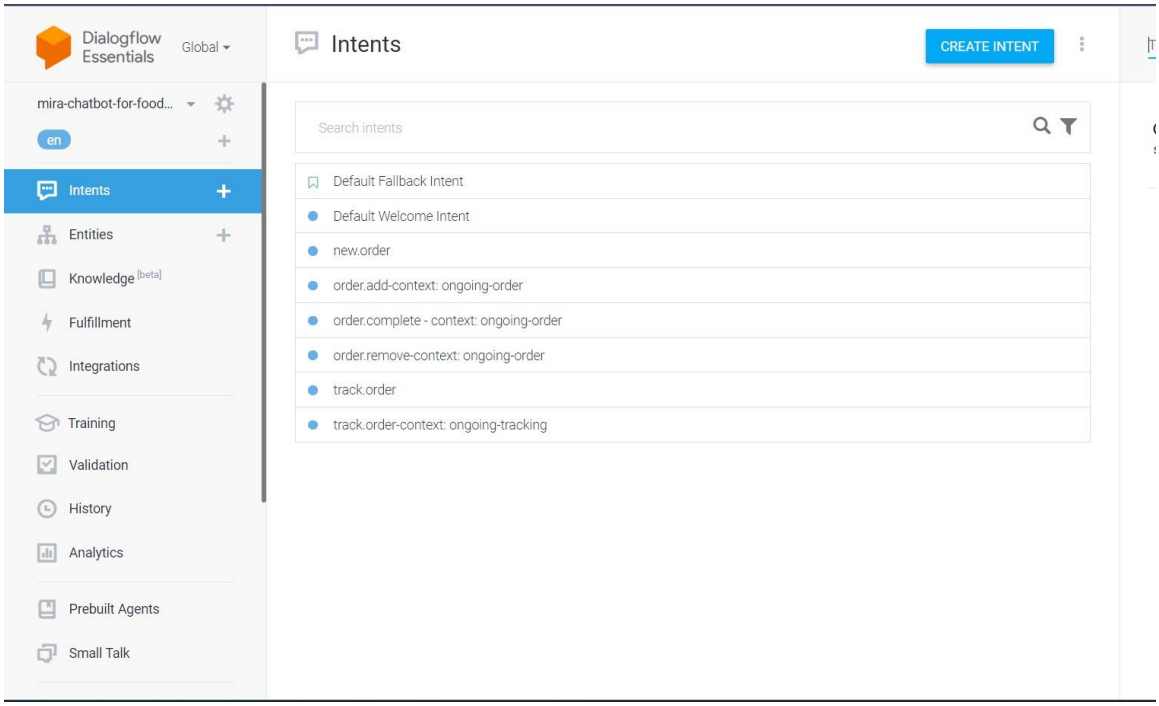
Dialogflow Essentials
Global
mira-chatbot-for-food...
en
Intents
Entities
Knowledge [beta]
Fulfillment
Integrations
Training
Validation
History
Analytics
Prebuilt Agents
Small Talk
Intents
CREATE INTENT
Search intents
Default Fallback Intent
Default Welcome Intent
new.order
order.add-context: ongoing-order
order.complete - context: ongoing-order
order.remove-context: ongoing-order
track.order
track.order-context: ongoing-tracking

Fig 3.3

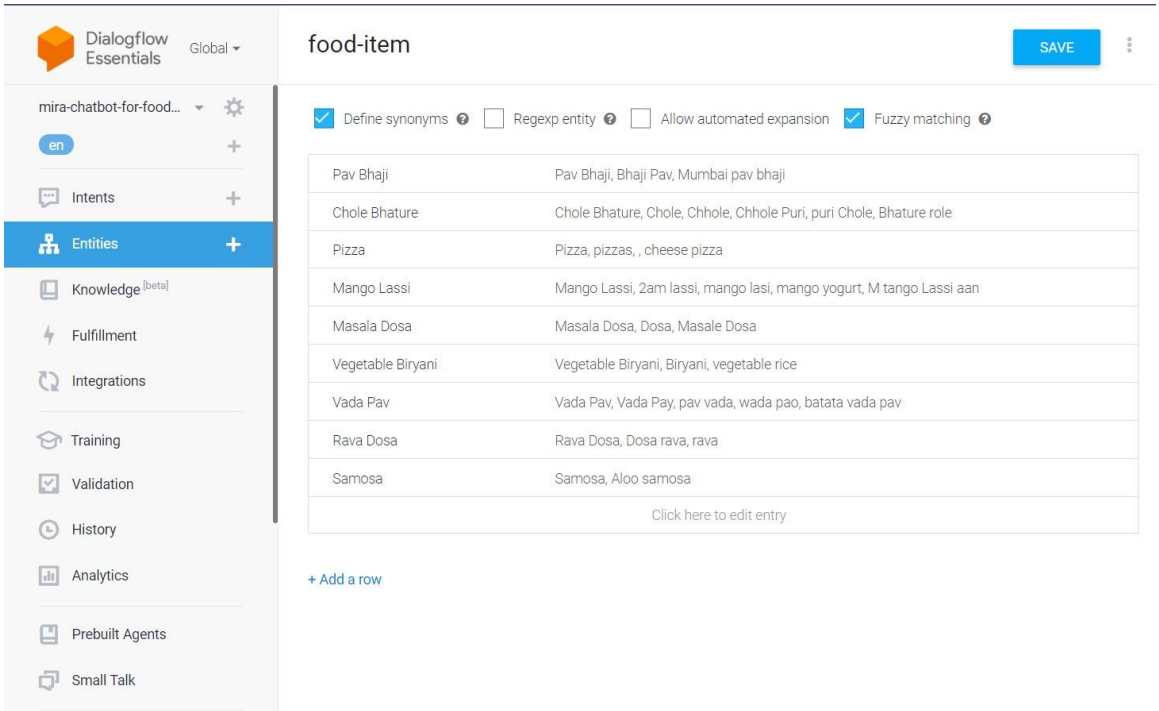
Dialogflow Essentials
Global
mira-chatbot-for-food...
en
Intents
Entities
Knowledge [beta]
Fulfillment
Integrations
Training
Validation
History
Analytics
Prebuilt Agents
Small Talk
food-item
SAVE
Define synonyms
Regexp entity
Allow automated expansion
Fuzzy matching
Pav Bhaji
Pav Bhaji, Bhaji Pav, Mumbai pav bhaji
Chole Bhature
Chole Bhature, Chole, Chhole, Chhole Puri, puri Chole, Bhature role
Pizza
Pizza, pizzas, , cheese pizza
Mango Lassi
Mango Lassi, 2am lassi, mango lasi, mango yogurt, M tango Lassi aan
Masala Dosa
Masala Dosa, Dosa, Masale Dosa
Vegetable Biryani
Vegetable Biryani, Biryani, vegetable rice
Vada Pav
Vada Pav, Vada Pay, pav vada, wada pao, batata vada pav
Rava Dosa
Rava Dosa, Dosa rava, rava
Samosa
Samosa, Aloo samosa
Click here to edit entry
+ Add a row

Fig 3.4

4. Results and Conclusion

The project has met its intended results as we have been able to create and successfully implement a Chatbot powered by NLP using DialogFlow and FastAPI. The project indeed, is not perfect in this form and may be subject to many revisions in the future to make a more marketable product.

5. References

[1] R. Garg *et al*., "NLP Based Chatbot for Multiple Restaurants," *2021 10th International Conference on System Modeling & Advancement in Research Trends (SMART)*, MORADABAD, India, 2021, pp. 439-443, doi: 10.1109/SMART52563.2021.9676218.

[2] H. Abdulla, A. M. Eltahir, S. Alwahaishi, K. Saghair, J. Platos and V. Snasel, "Chatbots Development Using Natural Language Processing: A Review," *2022 26th International Conference on Circuits, Systems, Communications and Computers (CSCC)*, Crete, Greece, 2022, pp. 122-128, doi: 10.1109/CSCC55931.2022.00030.

[3] K. Anjum, M. Sameer and S. Kumar, "AI Enabled NLP based Text to Text Medical Chatbot," *2023 3rd International Conference on Innovative Practices in Technology and Management (ICIPTM)*, Uttar Pradesh, India, 2023, pp. 1-5, doi: 10.1109/ICIPTM57143.2023.10117966