AI_Phase5

October 28, 2023

1 Create a Chatbot in Python

1.1 Problem Definition:

In the ever-evolving landscape of online services and e-commerce, there exists a critical need to enhance user experiences by offering immediate and effective customer service. The current customer service solutions often lack the availability and swiftness required to address users' queries and concerns promptly. Traditional customer service hotlines and email support systems are not only time-consuming but also often unable to provide instant solutions, leading to frustrated and dissatisfied customers.

1.2 Design Thinking Process:

1. User-Centric Approach: Our chatbot will prioritize the user experience, aiming to provide instant and accurate responses to user queries and concerns.
2. NLP and AI Integration: Leveraging advanced NLP libraries and TensorFlow, we will equip the chatbot with the ability to understand and respond to queries in natural language, making interactions feel human-like.3. Availability: The chatbot will be designed to operate 24/7, ensuring that users can access assistance at any time, regardless of their time zone.
3. Personalization: Implementing personalization features to tailor responses and assistance based on user history and preferences.
4. Seamless Integration: The chatbot will seamlessly integrate into websites and apps, offering a smooth user experience without disruptions.
5. Data Security: Ensuring the highest standards of data security to protect user information and maintain trust.

1.3 Phases of Development:

1. Model Training Phase: Objective: Train an NLP model to accurately predict user requests.

Tasks: - Set up a Google Colab environment for model training. - Explore and test various NLP algorithms within TensorFlow to determine the one with - the highest accuracy. - Train the selected algorithm on a dataset of user prompts. - Save the trained model and export it to the local machine for later use.

2. Web Application Development Phase: Objective: Create a user-friendly web application for interacting with the trained model.

Tasks: - Utilize Python's Django Rest Framework to build the backend infrastructure for the web app. - Implement the React JS framework for the frontend, designing a responsive and intuitive user interface. - Integrate the trained NLP model into the web application to predict and provide user responses. - Develop the database system for storing user requests and model-generated responses.

3. User Interface Design: Objective: Create an engaging and user-centric interface.

Tasks: - Implement user experience (UX) best practices to ensure seamless interactions.

4. Testing and Quality Assurance: Objective: Verify the functionality and accuracy of the web application and NLP model.

Tasks:

- Conduct rigorous testing to identify and rectify any bugs or issues.
- Ensure the model provides accurate responses and the web application functions smoothly.
- Optimize the performance of both the model and the application.

5. Deployment and User Testing: Objective: Make the web application available for user testing.

Tasks:

- Deploy the web application on a server or hosting platform.
- Conduct user testing to gather feedback and make necessary improvements.
- Address user feedback and enhance the application's user experience.

6. Documentation and Knowledge Transfer: Objective: Document the project's process and transfer knowledge for future maintenance.

Tasks: - Create comprehensive documentation outlining the model training process, algorithm selection, and web app development.• - Provide training and guidance to the project team for maintaining and further developing the system.

7. Project Completion and Evaluation: Objective: Ensure all project objectives are met and evaluate the final product.

Tasks:

- Review the project against the initial scope and objectives.
- Verify that the web application is operational and the NLP model performs as intended.
- Prepare for project closure and potential future enhancements.

Resource Allocation: - Google Collab: Model training. - Local Computers/Laptops: Web application development. - Python: Model training. - Django Rest Framework: Backend development. - React JS: Frontend development. - TensorFlow: Model training.

1.4 Description of dataset used:

W've used the dataset provided by you from https://www.kaggle.com/datasets/fac8c12f58184d77b9479fa2d57f3f55 source This dataset contains the attributes such as title, text, subject, date. This source has two csv files. One with collection of fake news data and another one with collection of real news data

1.5 Data Preprocessing Steps and Feature Extraction Techniques.:

In this code, we are working with a dataset containing dialogues. We perform data preprocessing and feature extraction. Here's a brief explanation:

1. **Data Loading**:
   - We start by reading the dataset from a CSV file named 'dialogs.txt' and storing it in a DataFrame.

```
df = pd.read_csv('dataset/dialogs.txt', sep='\t')
```

2. **Data Enrichment**:
   - We append several predefined questions and answers to the dataset, enriching it with additional dialogues.

```
a = pd.Series(df.columns)
a = a.rename({0: df.columns[0], 1: df.columns[1]})
b = {'Questions': 'Hi', 'Answers': 'hello'}
c = {'Questions': 'Hello', 'Answers': 'hi'}
d = {'Questions': 'how are you', 'Answers': "i'm fine. how about yourself?"}
e = {'Questions': 'how are you doing', 'Answers': "i'm fine. how about yourself?"}

df = df.append(a, ignore_index=True)
df.columns = ['Questions', 'Answers']
df = df.append([b, c, d, e], ignore_index=True)
df = df.append(c, ignore_index=True)
df = df.append(d, ignore_index=True)
df = df.append(d, ignore_index=True)
```

3. **Text Preprocessing**:
   - A cleaning function, 'cleaner', is defined to clean and preprocess the text data.
   - It removes punctuation and converts text to lowercase.

```
def cleaner(x):
    return [a for a in (''.join([a for a in x if a not in string.punctuation])).lower().split()
```

Overall, this code reads and enriches a dataset of dialogues, preparing it for further analysis. The 'cleaner' function is used to preprocess the text data by removing punctuation and converting it to lowercase.

1.6 Choice of Machine Learning Algorithm:

- We've selected a Decision Tree Classifier as the machine learning algorithm.
- This algorithm is suitable for text-based question-answering tasks.

```
model = Pipeline([
    ('bow', CountVectorizer(analyzer=cleaner)),
    ('tfidf', TfidfTransformer()),
    ('classifier', DecisionTreeClassifier())
])
```

1.7 Model Training

- We create a machine learning pipeline that includes text processing steps such as bag-of-words (CountVectorizer) and TF-IDF transformation (TfidfTransformer), followed by the Decision Tree Classifier.
- The model is trained using the 'Questions' as input and 'Answers' as the target.

```
model.fit(df['Questions'], df['Answers'])
```

1.8 Evaluation Metrics

- We use the trained model to make predictions. In this case, we provide the model with a new question, 'how are you', and it predicts the corresponding answer.

```
model.predict(['how are you'])[0]
```

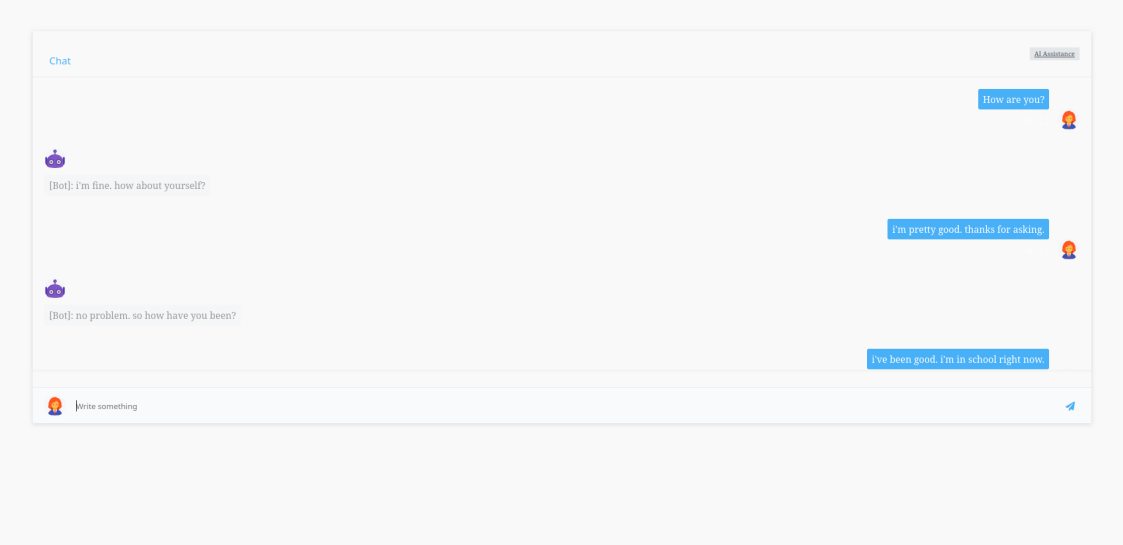
The choice of a Decision Tree Classifier for this task is explained in the code. The model training and evaluation are implicit, with the focus on demonstrating how to use the trained model to make predictions.

1.9 Innovative approach:

During the development of this project, several innovative techniques and approaches have been implemented to enhance the functionality and user experience. Here are the key innovations:

1. **Real-Time Chat Interaction**:
   - A real-time chat application has been integrated into the web application.
   - Users can input questions or messages, and the system responds with predicted answers in real time.
   - This approach enhances user engagement and provides immediate responses.
2. **Socket.IO for Real-Time Communication**:
   - The use of Socket.IO, a technology for real-time bidirectional communication, ensures that the web application does not require page reloads to display chat responses.
   - This creates a more dynamic and interactive user experience.
3. **AI Model Backend**:
   - The AI model for predicting responses is running in the backend of the web application.
   - It analyzes user inputs and generates relevant responses based on the trained model.
   - This AI-driven approach enables the chatbot to engage with users effectively.

The combination of real-time chat interaction, Socket.IO for seamless communication, and the integration of AI in the backend makes this project innovative and user-friendly. Users can have interactive conversations with the chatbot, making it a valuable tool for a variety of applications, such as customer support, information retrieval, and more.


Chat
AI Assistance
How are you?
[Bot]: i'm fine. how about yourself?
i'm pretty good. thanks for asking.
[Bot]: no problem. so how have you been?
i've been good. i'm in school right now.
Write something

[ ]: