Chatbot Machine Learning Project

Ghate Nikhita

Project Overview

- Goal: Develop a chatbot capable of interacting with users in the Customer Support domain.
- Domain: Customer Support, focusing on 27 intents grouped into 11 categories such as refunds, orders, invoices, and accounts.
- Use Cases:
  - Provide quick support by recognizing user intents.
  - Automate responses for frequently asked questions (FAQs).
  - Enhance customer service efficiency by handling a variety of inquiries.

Dataset Overview

01. Source: Bitext Sample Customer Service Training Dataset.

02.
- Size:
- Over 20,000 utterances.
- 27 intents, categorized into 11 groups.

03. Linguistic Features:
- Includes spelling mistakes, punctuation errors, colloquialisms, and polite vs. offensive language.
- Annotations for linguistic phenomena (Basic structure, Politeness, Offensive language, etc.) to better mimic real-life interactions.

Categories & Intents

- Categories: ACCOUNT, REFUNDS, SHIPPING, DELIVERY, PAYMENT, etc.
- Examples of Intents:
- cancel_order: Users requesting to cancel an order.
- track_order: Users inquiring about the status of their orders.
- payment_issue: Problems with payments, billing, and invoices.

Natural Language Processing (NLP)

01

Intent Recognition: Classify user utterances into predefined intents (e.g., track_order, get_refund).

02

Utterance Cleaning:

- Remove special characters, punctuation.
- Normalize text (lowercasing, stopword removal).

03

- Feature Extraction:
- Use TF-IDF (Term Frequency-Inverse Document Frequency) to convert textual data into numerical vectors.

04

Label Encoding: Convert categorical intent labels into machine-readable formats.

Machine Learning Model

Retrieval-Based Model:

- Uses cosine similarity to match user queries with the most similar pre-defined utterances.
- TF-IDF Vectorization: Converts both user queries and training utterances into TF-IDF matrices.
- Cosine Similarity: Finds the best matching response by calculating the similarity between vectors.

Example of Functionality:

Test Query: “Could I check if there is anything wrong with my refund?
Predicted Intent: track_refund

Training & Testing Data

Data Split: 80% training, 20% testing.

Training set size: 6540+ utterances.

Testing set size: 1635+ utterances.

Tokenization & Vectorization: Apply TF-IDF to convert textual data into vectors.

Train machine learning models using this numerical data to classify intents.

Model Evaluation & Metric

- Key Metrics:
- Accuracy: How well the model predicts the correct intent.
- Precision & Recall: Important for ensuring that correct intents are predicted and irrelevant responses are minimized.
- Confusion Matrix: Visual representation of where the model correctly or incorrectly classifies intents.

Challenges & Linguistic Considerations

- Real-life Phenomena:
- Handling spelling mistakes, missing punctuation, and offensive language.
- Managing variations in politeness or slang (e.g., “I’d like to cancel my order” vs. “Cancel order now!”).
- Multiple Utterance Lengths: Some intents have shorter utterances while others involve more complex questions.

Deployment Plan

- Web Application: Deploy chatbot using Flask, allowing users to interact with the model in real-time.

- Integration:
- Front-end with HTML/CSS templates.
- Interaction through simple chat interfaces.

- Real-time Predictions: Users will type queries, and the chatbot will return predicted intents based on the trained model.

Future Enhancements

- Handling Multiple Languages: Expand chatbot to support different languages using additional datasets.
- Improving Accuracy: Use more advanced techniques like deep learning models (e.g., RNNs or Transformers).
- Contextual Understanding: Implementing context-awareness to handle more complex multi-turn conversations.

Demo

```python
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import numpy as np

# Fit the TF-IDF vectorizer on the training data
vectorizer = TfidfVectorizer(max_features=3000)
X_train_tfidf = vectorizer.fit_transform(X_train).toarray()

# Define a function to retrieve the most similar response
def get_response(query):
    # Transform the user query to TF-IDF
    query_tfidf = vectorizer.transform([query]).toarray()

    # Compute cosine similarity between the query and all training utterances
    similarities = cosine_similarity(query_tfidf, X_train_tfidf)

    # Get the index of the most similar utterance
    max_sim_index = np.argmax(similarities)

    # Return the corresponding response (intent or action)
    return y_train.iloc[max_sim_index]

# Test the retrieval-based model
query = "could I check if there is anything wrong with my refund?"
response = get_response(query)
print(f"User Query: {query} \nPredicted Response/Intent: {response}")
```

✓ 0.5s

```
User Query: could I check if there is anything wrong with my refund?
Predicted Response/Intent: track_refund
```