“NLP-Enhanced Customer Service Expert System”

Rahul Bandi

CS 52400 – 01 Expert Systems

Dr. Mohammadreza Hajiarbabi

Table of Contents

1. Introduction

In the situation of rapid evolution of artificial intelligence improving customer service has become a key goal for companies around the world. **"NLP Improved Customer Service Expert System"** leverages the transformative power of Natural language processing (NLP) and deploys expert systems to define new customer support experience. This project addresses inefficiencies of traditional systems, such as slow response times, Limited customization and inability to handle complex queries, by combining state-of-the-art technologies such as **BERT** (Bidirectional Encoder Representations from Transformers), Rule-based reasoning, and OpenAI's GPT-3.5.

This innovative approach combines the predictive resources of machine learning with the interpretive capabilities of humans. It offers a dynamic, scalable, and easy-to-use Chabot solution. By automatically classifying questionnaires Improving interpretation through rules and the use of advanced language models for ambiguous cases. This system creates a new level of efficiency and reliability in customer support.

Key Features:

1. **Predictive Power of BERT**
    - The core of the system is a fine-tuned BERT model, designed to classify customer queries accurately.
    - Pre-trained on vast text data and fine-tuned on domain-specific customer service datasets, the model achieves impressive validation accuracy and F1 scores, ensuring robust intent recognition.
2. **Rule-Based Reasoning**
    - Augments machine learning by introducing predefined rules and confidence-based logic.
    - Ensures interpretability and reliability by mapping customer queries to specific intents based on keywords and confidence thresholds.
3. **OpenAI Integration**
    - Incorporates GPT-3.5 to handle ambiguous or edge-case queries effectively.
    - Dynamically generates fall-back responses, ensuring users always receive meaningful interactions, even in uncertain scenarios.
4. **Interactive User Interface**
    - Powered by FastAPI and HTML/CSS, the Chabot offers a seamless and intuitive experience.

- Features include real-time query handling, dynamic confidence score displays, and an interactive testing environment for customers and developers.

5. **Scalability and Adaptability**

    - Modular architecture allows for easy integration of new intents, additional languages, and voice-based inputs.

    - Designed to cater to a wide range of industries, from e-commerce and banking to healthcare and telecommunications.

6. **Efficient Query Handling**

    - Automates the classification and resolution of high-volume customer queries, significantly reducing response times and operational costs.

7. **Data-Driven Insights**

    - Provides actionable insights into customer behaviour and query patterns, enabling businesses to refine their services continually.

Combining the predictive power of machine learning Interpretation of rules-based systems and the dynamic adaptability of language models, this system offers a robust solution to modern customer service challenges. It represents a shift towards intelligent automation that balances efficiency with a human-centred touch. Ultimately increasing customer satisfaction and loyalty.

2. Motivation

Traditional customer support systems often face challenges such as slow response times, inability to handle complex queries, and lack of personalization. This project was inspired by these constraints and the goal is to develop a system that combines machine learning and expert systems for:

- Automatically sorts customer inquiries i.e., classification
- Provide real-time answers with high accuracy.
- Enhance user satisfaction through personalized and dynamic interactions.

Real-World Impact:
Automating customer service processes provides the following benefits:

- **Time Savings**: Faster response times reduce the waiting period for customers.
- **Operational Efficiency**: Automation minimizes human intervention, cutting operational costs.
- **Improved Customer Experience**: Personalized and accurate responses enhance customer satisfaction and loyalty.
- **Scalability**: The system can handle high volumes of queries, making it suitable for businesses of all sizes.

3. Questions That Guided This Project

3.1 What are the common intents in customer support queries?

Understanding the typical reasons for customer interactions is the foundation of creating an efficient support system. To identify these intents, historical customer support datasets were analysed. The data revealed common themes such as:

- **Refund Requests**: Queries about getting a refund for a purchase or resolving payment disputes.
- **Technical Issues**: Complaints about app crashes, connectivity issues, or feature malfunctions.
- **Account Access Problems**: Concerns about forgotten passwords, locked accounts, or updating account details.
- **General Inquiries**: Questions about product availability, service updates, or company policies.

Identifying these intents allows the system to predefine categories, simplifying the classification process and enabling precise responses.

3.2 How to ensure accuracy and reliability of responses?

To build user trust, the system must provide accurate and relevant responses consistently. This was achieved by:

- **Fine-Tuning BERT**: Leveraging BERT's contextual understanding capabilities to classify intents with high precision. Fine-tuning involved training the model on 200,000 labelled queries to adapt it to the nuances of customer service language.
- **Confidence Scoring**: Adding rule-based confidence adjustments, such as considering the frequency and position of key terms in a query.
- **Fallback Mechanisms**: When the system encounters ambiguity, OpenAI’s GPT-3.5 is used to generate dynamic, contextually relevant responses.

This multi-layered approach ensures that the system is robust and reliable, providing clear answers to structured queries while gracefully handling ambiguous ones.

3.3 How to combine rule-based systems with machine learning for a seamless experience?

This project bridges traditional expert systems and modern machine learning techniques through:

- **Rule-Based Reasoning**: A set of predefined rules maps specific keywords and patterns to intents with adjustable confidence thresholds.

- **BERT Predictions**: BERT’s predictions enhance the rule-based system by offering a deeper contextual understanding of queries.

- **Dynamic Integration**: For queries where rules and BERT predictions are insufficient, OpenAI’s GPT-3.5 dynamically generates fallback responses, maintaining the flow of interaction.

4. Methodology

The system is designed for accurate intent classification, robust reasoning, and an interactive user interface to address customer queries effectively. This System for the NLP-Enhanced Customer Service Expert System integrates state-of-the-art Natural Language Processing (NLP) models, rule-based reasoning, and dynamic response generation using OpenAI's GPT-3.5

4.1 Knowledge Base

The knowledge base acts as the foundational element, encompassing structured data, predefined rules, and intent mapping strategies:

4.1.1. Dataset:

- **Source:** The dataset consists of subset of Twitter Data over 200,000 real-world customer support queries sourced from platforms like Kaggle and proprietary datasets.

- **Structure:** Each record includes:

    - **text:** The actual customer query.
    - **inbound:** A boolean flag distinguishing user messages from agent responses.
    - **intent labels:** Categories such as "refund_request," "technical_issue," and "account_access."

- **Preprocessing**

    - **Text Cleaning:** Removal of noise such as URLs, mentions, and special characters using Python’s re module.
    - **Duplication Removal:** Ensured unique records by eliminating duplicates.
    - **Balancing:** Oversampled minority intents to address class imbalance.

4.1.2 Intent Mapping Rules:

- Keywords and patterns were defined for each intent.
- Confidence weights assigned to intents based on keyword presence, text length, and query complexity.

Example:

Rule: If the query contains "refund" or "money back" and is inbound, classify the intent as "refund_request."

4.2 Inference Engine

The inference engine combines machine learning with rule-based reasoning for robust intent classification and response generation:

4.2.1 BERT Fine-Tuning

- **Model**: Pre-trained bert-base-uncased model from Hugging Face.
- **Training**:
  - **Objective**: Multi-class classification to predict customer intent.
  - **Setup**:
    - Learning rate: 2e-5
    - Batch size: 32
    - Epochs: 3
  - **Dataset Split**:
    - 80% for training.
    - 20% for validation.
  - **Fine-Tuning Process**:
    - Layer weights adjusted for specific customer support intents.
    - Used GPU acceleration to optimize training time.
- **Performance**:
  - **Validation accuracy:** 89.6%
  - **F1-score:** 89.3%
  - Precision/recall trade-offs optimized using weighted metrics.

4.2.2 Rule-Based Reasoning

Rule-based reasoning is a logical approach within expert systems where a set of predefined rules is applied to make decisions or infer conclusions based on input data. These rules are crafted to reflect domain-specific knowledge and provide deterministic responses when certain conditions are met.

In the context of the **NLP-Enhanced Customer Service Expert System**, rule-based reasoning complements the BERT model by ensuring interpretable and predictable outcomes, especially for edge cases or queries with low machine learning confidence. It bridges the gap between probabilistic model outputs and deterministic expert knowledge.

Key components of rule-based reasoning in this project:

1. **Predefined Rules**: Created based on common intents and patterns in customer queries.
2. **Confidence Thresholds**: Ensure only high-confidence rules are triggered, enhancing accuracy.
3. **Fallback Handling**: Low-confidence cases are delegated to GPT-3.5 for dynamic responses.

4.2.2.1 List of Rules Used in the System

The following **rules** are inferred from the **knowledge base** provided. These rules map user queries to specific intents and their corresponding responses.

Intent Mapping Rules:

1. **Refund Request**
    - **If**: Query contains keywords like "refund" or "money back".
    - **Then**: Intent is classified as refund_request.
2. **Refund Status**
    - **If**: Query contains keywords like "refund status" or "status of my refund".
    - **Then**: Intent is classified as refund_status.
3. **Technical Issue**
    - **If**: Query contains keywords like "issue", "error", "problem", "crash", or "not working".
    - **Then**: Intent is classified as technical_issue.
4. **App Crash**
    - **If**: Query contains keywords like "app crash" or "application stopped working".
    - **Then**: Intent is classified as app_crash.
5. **Connectivity Issue**
    - **If**: Query contains keywords like "internet", "connection issue", or "unable to connect".
    - **Then**: Intent is classified as connectivity_issue.

6. **Password Reset**

    - **If**: Query contains keywords like "forgot password", "reset password", or "change password".
    - **Then**: Intent is classified as password_reset.

7. **Account Locked**

    - **If**: Query contains keywords like "account locked", "cannot log in", or "access denied".
    - **Then**: Intent is classified as account_locked.

8. **Account Details Update**

    - **If**: Query contains keywords like "update details", "change email", or "update profile".
    - **Then**: Intent is classified as account_details_update.

9. **Product Availability**

    - **If**: Query contains keywords like "in stock", "available", or "when will it be available".
    - **Then**: Intent is classified as product_availability.

10. **Product Not Received**

    - **If**: Query contains keywords like "not received", "missing product", or "delivery issue".
    - **Then**: Intent is classified as product_not_received.

11. **Return Product**

    - **If**: Query contains keywords like "return product", "how to return", or "refund for product".
    - **Then**: Intent is classified as return_product.

12. **Billing Issue**

    - **If**: Query contains keywords like "billing", "incorrect charge", "payment issue", or "charged incorrectly".
    - **Then**: Intent is classified as billing_issue.

13. **Invoice Request**

    - **If**: Query contains keywords like "invoice", "receipt", or "billing statement".
    - **Then**: Intent is classified as invoice_request.

14. **Positive Feedback**

- **If**: Query contains keywords like "good service", "thank you", or "great experience".
- **Then**: Intent is classified as positive_feedback.

15. **Negative Feedback**

- **If**: Query contains keywords like "bad service", "terrible experience", or "disappointed".
- **Then**: Intent is classified as negative_feedback.

16. **Escalate Issue**

- **If**: Query contains keywords like "escalate", "manager", or "higher authority".
- **Then**: Intent is classified as escalate_issue.

17. **Speak to Agent**

- **If**: Query contains keywords like "speak to an agent", "connect to support", or "human support".
- **Then**: Intent is classified as speak_to_agent.

4.2.2.2Logic Implementation of Rule-Based Reasoning

1. **Rule Evaluation**:

   - For each incoming query, the system evaluates all rules sequentially.
   - If multiple rules match, the rule with the highest confidence is selected.
   - Confidence is further adjusted based on:

     - Query length: Shorter queries reduce confidence.
     - Keyword proximity: Earlier occurrences of keywords increase confidence.

2. **Example Workflow**:

   - Query: "My app crashes when I try to open it."
   - Matching Rule: Technical Issue.
   - Base Confidence: 0.7.
   - Adjusted Confidence: 0.72 (based on query length and keyword position).
   - Result: Intent classified as technical_issue.

3. **Confidence Threshold**:

   - A confidence score of 0.5 or above is required for a rule to trigger.
   - If no rule meets the threshold, the fall-back mechanism invokes GPT-3.5.

4. **Dynamic Updates**:

    - The system updates rules dynamically based on performance evaluations, ensuring adaptability to new customer queries or intents.

4.2.3 OpenAI API Integration

- **Dynamic Response Generation**:

    - OpenAI's GPT-3.5 ChatCompletion API enhances responses for queries not covered by predefined intents.

        Example: When BERT predicts an intent with low confidence, GPT-3.5 generates a detailed and polite fall-back response.

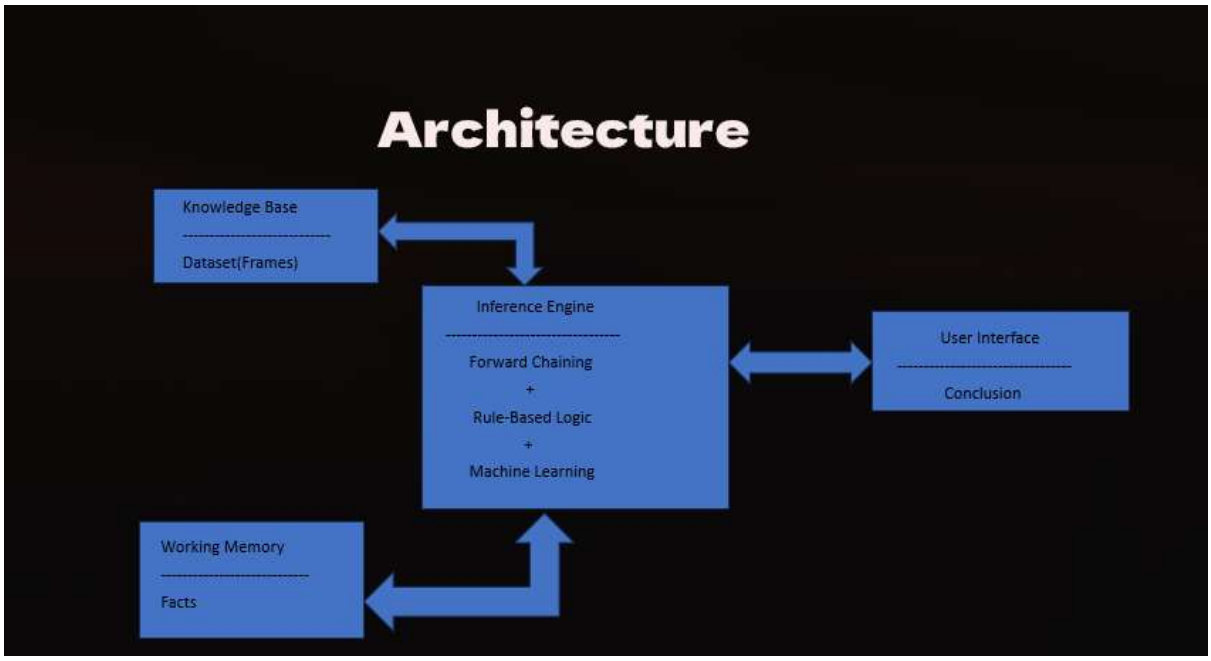
    - **API Advantages**:
        - Handles conversational ambiguities.
        - Provides empathetic and human-like responses.

- **Integration Strategy:**

    - Combined GPT-3.5 outputs with the rule-based system to ensure seamless user experiences.

    - Reduced reliance on the model for repetitive or simple queries, offloading these to predefined rules.

Architecture



4.3 User Interface

The user interface (UI) bridges the system and the user, ensuring accessibility and real-time interaction:

- **Platform**:
    - Built using FastAPI for the backend and HTML/CSS for the frontend.
    - Deployed locally via Uvicorn and exposed globally using Ngrok.
- **Design**:
    - Minimalist, intuitive, and user-friendly.
    - Features a chatbox that dynamically updates with user queries and system responses.
- **Core Features**:
    - **Real-Time Interaction:**
        - Immediate display of predictions and responses.
    - **Confidence Scores:**
        - Transparent display of the model's confidence in intent classification.
        - Enhances user trust by revealing system reasoning.
    - **Deployment:**
        - Integrated API endpoints for seamless user input processing.
        - Ensured scalability for handling multiple concurrent users.
- **Visual Enhancements**:
    - Dynamic response highlighting for better readability.
    - Smooth scrolling and user-friendly text input options.

5. Challenges

The development of the **NLP-Enhanced Customer Service Expert System** faced several challenges across various stages, ranging from data preparation to dynamic response generation. These challenges, along with the strategies employed to address them, are detailed below:

5.1. Data Preparation

5.1.1. Imbalanced Dataset

Challenge:

- The dataset contained significant disparities in the distribution of intents. For example, intents like "refund requests" were overrepresented, while others such as "service complaints" or "billing issues" had fewer instances.
- Machine learning models often struggle with imbalanced datasets, leading to biased predictions toward majority classes.

Solution:

- Oversampling Minority Intents:
    - Oversampling techniques were applied to replicate data points from underrepresented intents, creating a balanced dataset.
    - This ensured that all intents were adequately represented during training.

Stratified Sampling:

- Ensured consistent representation of all intents in training and validation datasets.

5.1.2. Text Cleaning

Challenge:

- Customer queries often contained noise such as:
    - URLs (e.g., "visit **http://example.com** for details").
    - Mentions and hashtags (e.g., "@user, #support").
    - Unnecessary white spaces and special characters.
- This noise could hinder the model's ability to accurately classify intents.

Solution:

- **Regex-Based Cleaning:**
    - Removed URLs, mentions, and hashtags using regular expressions.

- **Normalization:**
    - Converted all text to lowercase and standardized whitespace to improve model consistency.
- **Domain-Specific Stopword Removal:**
    - Removed non-informative words specific to the domain (e.g., "please," "thanks").

5.2. Computational Limitations

5.2.1. GPU Constraints

- BERT fine-tuning is computationally intensive and required substantial GPU resources.
- The model's large size led to memory bottlenecks, especially during batch processing.

Solution for Above issue:

- **Optimization of Training Pipeline:**
    - Used gradient accumulation to simulate larger batch sizes without exceeding GPU memory limits.
    - Enabled mixed-precision training (FP16) to reduce memory usage.
- **Resource Allocation:**
    - Leveraged cloud-based GPU solutions for scaling during training.

5.2.2. Hyperparameter Tuning

Challenge:

- Selecting optimal values for hyperparameters such as learning rate, batch size, and number of epochs was critical to achieving high performance.
- A poorly tuned model could lead to underfitting or overfitting.

Solution:

- Conducted systematic hyperparameter search using validation accuracy and F1-score as metrics.
- Monitored performance during training to halt when no significant improvement was observed.

5.3. Dynamic Response Generation

5.3.1. Handling Ambiguities

Challenge:

- Customer queries are not always straightforward. Ambiguous queries can fall between multiple intents or lack enough context for rule-based or ML classification.

Solution with Integration of OpenAI API:

- For ambiguous queries, GPT-3.5 was used to generate dynamic responses.
- This provided flexibility and enhanced user interaction for edge cases.

6. Future Scope

Multilingual Support

- Expand the knowledge base with multilingual datasets.
- Fine-tune BERT models like mBERT (multilingual BERT) or XLM-RoBERTa to support diverse languages.
- Integrate language detection tools to route queries to appropriate language-specific pipelines.

Voice Integration

- Integrate speech-to-text (STT) models for transcribing voice queries (e.g., Google Speech API or Whisper).
- Combine with existing NLP pipelines to process transcribed text.
- Use text-to-speech (TTS) models to generate spoken responses, creating a fully interactive experience.

Fuzzy Logic

- Implement fuzzy logic to assign degrees of truth to rule-based confidence scores.
- This would improve the system's ability to handle edge cases and ambiguous queries.

7. References

1. Lecture Notes.
2. "Natural Language Processing in Customer Service: A Systematic Review" by Malak Mashaabi, Areej Alotaibi, Hala Qudaih, Raghad Alnashwan, Hend Al-Khalifa
3. "NLP techniques for automating responses to customer queries: a systematic review" by Peter Adebowale Olujimi1 · Abejide Ade-Ibijola2
4. Expert systems: design and development by Durkin, John. Macmillan Publishing Company (1994).
5. Hugging Face Transformers Documentation.
6. openai API Documentation.
7. Kaggle Dataset for Customer Support Queries.