

**SAVEETHA SCHOOL OF ENGINEERING**

**SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES**

# CAPSTONE PROJECT REPORT

**PROJECT TITLE**

Enhancing Customer Support with a Context-Aware Chatbot: Leveraging NLP Techniques and Rasa Framework

# REPORT SUBMITTED BY

192210286 G.vivek

192210020 Brama Rao

**REPORT SUBMITTED TO**

Dr. E. MONIKA

# COURSE CODE / COURSE NAME

CSA1369 / THEORY OF COMPUTATION WITH PROBLEM SOLVING

SLOT C

# DATE OF SUBMISSION

11.09.2024

# ABSTRACT

This project focuses on the design and implementation of a context-aware chatbot tailored for customer support in a specific domain, such as e-commerce or healthcare. The chatbot is built using advanced Natural Language Processing (NLP) techniques to accurately understand user queries, manage multi-turn conversations, and deliver precise, contextually relevant responses. By incorporating intent recognition and entity extraction, the chatbot can effectively categorize user requests and identify key information within the conversation. Additionally, the system includes an escalation mechanism that seamlessly transitions complex queries to human agents, ensuring that users receive appropriate support when needed.The chatbot integrates a domain-specific knowledge base, allowing it to provide detailed and accurate responses that are directly relevant to the industry it serves. The project employs tools and technologies such as Python, TensorFlow, PyTorch, and Rasa to develop and fine-tune the underlying models. Performance evaluation of the chatbot is conducted using metrics such as accuracy, precision, recall, and F1 score, with the results demonstrating the chatbot's capability to enhance customer support efficiency and improve user satisfaction. This project highlights the potential of NLP-driven conversational bots in transforming customer service experiences by reducing response times and improving the accuracy of automated support.

# INTRODUCTION

# Conversational bots, commonly known as chatbots, have become a critical component of modern customer service operations across various industries. With the rapid growth of digital platforms, businesses are increasingly relying on chatbots to provide immediate and efficient responses to customer inquiries. These bots use Natural Language Processing (NLP) to simulate human-like conversations, offering users a seamless way to access information, resolve issues, and complete transactions. The ability of chatbots to operate 24/7 and handle multiple inquiries simultaneously makes them indispensable in environments where timely customer support is essential.

# The applications of chatbots extend beyond customer service into areas such as healthcare, banking, and e-commerce. In healthcare, chatbots can assist patients by providing information on symptoms, medication reminders, and appointment scheduling. In banking, they can handle queries related to account balances, transactions, and financial advice. E-commerce platforms use chatbots to assist customers with product recommendations, order tracking, and returns. By integrating chatbots with domain-specific knowledge bases, businesses can provide personalized and accurate responses, enhancing the overall user experience and improving customer satisfaction.

# The advantages of using chatbots are numerous. They significantly reduce the operational costs associated with customer support by automating routine tasks and minimizing the need for human intervention. Chatbots also improve response times, ensuring that customers receive prompt assistance, which can lead to higher customer retention rates. Furthermore, advanced chatbots that manage conversation context and handle multi-turn dialogues contribute to a more engaging and effective interaction, making customers feel heard and valued. As technology continues to evolve, the role of chatbots in enhancing business operations and customer relations is expected to expand further.

Moreover, the evolution of artificial intelligence and machine learning has greatly enhanced the capabilities of chatbots, making them more intuitive and sophisticated. Modern chatbots are not only able to understand and respond to simple queries, but they can also manage complex interactions that involve multiple turns of conversation. By leveraging pre-trained language models and advanced NLP techniques, these bots can maintain context, recognize user intent, and extract relevant entities from the dialogue. This progression from basic rule-based systems to context-aware conversational agents represents a significant leap forward in how businesses can engage with their customers, providing a more personalized and dynamic support experience.

# RESEARCH PLAN

The research plan for the "Context-Aware Chatbot for Customer Support" project is structured into five key phases. The first phase, Project Initiation and Planning, involves defining the project’s scope and objectives, identifying stakeholders, and establishing a comprehensive project plan with timelines and resource allocation. Next, in the Data Collection and Preprocessing phase, relevant customer service data is gathered from various sources and processed using NLP techniques, including steps such as tokenization, stopword removal, and lemmatization, ensuring that the text data is clean and suitable for chatbot training. The third phase, Development and Implementation, focuses on building the chatbot using pre-trained NLP models for intent recognition, entity extraction, and context management. This phase also includes integrating the chatbot with a domain-specific knowledge base and implementing an escalation mechanism to route complex queries to human agents. In the Testing and Evaluation phase, the chatbot undergoes rigorous testing, including unit tests, integration tests, and user acceptance testing, to ensure it accurately and reliably handles customer queries. Finally, the Documentation, Deployment, and Feedback phase involves documenting the development process, preparing the chatbot for deployment, and gathering user feedback to refine and enhance the system. This structured approach ensures a thorough and effective implementation of a context-aware chatbot for customer support.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| S.NO | DESCRIPTION | 04.09.2024  DAY-01 | 05.09.2024  DAY-02 | 06.09.2024  DAY-03 | 09.09.2024  DAY-04 | 10.09.2024  DAY-05 |
| 1. | Project Initiation and Planning |  |  |  |  |  |
| 2. | Requirement Analysis and Design |  |  |  |  |  |
| 3. | Development and Implementation |  |  |  |  |  |
| 4. | Testing and Refinement |  |  |  |  |  |
| 5. | Documentation, Deployment, and Feedback |  |  |  |  |  |

**Fig. 1 Timeline chart**

**Day 1: Project Initiation and Planning (1 day)**

* **Scope and Objectives:** Define the scope and objectives of the project, focusing on developing a context-aware chatbot for customer support. Identify key goals such as improving customer interaction accuracy, enhancing response relevance, and ensuring smooth multi-turn conversations.
* **Initial Research:** Conduct initial research on chatbot design, including NLP techniques for intent recognition, entity extraction, and context management. Review best practices for integrating chatbots with domain-specific knowledge bases and escalation mechanisms.
* **Stakeholder Identification:** Identify key stakeholders, including developers, project sponsors, customer support teams, and potential end-users. Establish communication channels for effective collaboration. Develop a detailed project plan with tasks and milestones for each subsequent phase of the project.

**Day 2: Requirement Analysis and Design (1 day)**

* **Data Collection:** Gather relevant customer service data for chatbot training, including chat transcripts and frequently asked questions (FAQs) from existing customer support interactions.
* **Preprocessing Implementation:** Implement text preprocessing steps using NLP libraries such as NLTK, including tokenization, stopword removal, and lemmatization, to ensure that the data is clean and ready for chatbot training.
* **Pipeline Finalization:** Finalize the preprocessing pipeline and verify its effectiveness in preparing the text data for accurate intent recognition and context management.

**Day 3: Development and Implementation (2 days)**

* **Core Functionality Development:** Develop the core functionality of the chatbot, focusing on implementing NLP models for intent recognition and entity extraction, as well as context management to handle multi-turn dialogues. Utilize libraries such as Rasa, TensorFlow, or PyTorch for efficient NLP model development.
* **Feature Integration:** Code and test the chatbot’s core functionalities, ensuring it can handle various customer queries and maintain conversation context. Integrate additional features such as an escalation mechanism to route complex queries to human agents.
* **System Testing:** Conduct initial testing of the chatbot’s functionality, verifying the accuracy of intent recognition and the chatbot’s ability to manage conversation context effectively.

**Day 4: Testing and Refinement (1 day)**

* **Comprehensive Testing:** Conduct comprehensive testing, including unit tests for individual chatbot components, integration tests to ensure smooth interaction across the system, and user acceptance testing to validate the chatbot’s effectiveness in real-world scenarios.
* **Bug Resolution and Feedback:** Identify and resolve any issues or bugs discovered during testing. Gather feedback from initial users to address usability concerns and performance issues.
* **System Refinement:** Refine the chatbot based on testing results and feedback to ensure that it meets the project’s objectives and performs reliably in customer support scenarios.

**Day 5: Documentation, Deployment, and Feedback (1 day)**

* **Documentation:** Document the entire development process, including methodologies for NLP model implementation, text preprocessing, and conversation management, as well as the testing and evaluation procedures.
* **Deployment Preparation:** Prepare the chatbot for deployment, ensuring it is configured for optimal performance and usability.
* **Final Deployment and Feedback:** Deploy the chatbot in a testing environment for final validation and gather feedback from stakeholders and end-users to assess its effectiveness in handling customer support queries. Evaluate the project’s success in achieving its objectives and identify areas for future improvement.

# MATERIALS AND METHODS

**1. Materials**

* **Hardware:**
  + **Development Machine:** A computer with at least 8GB RAM, multi-core processor, and a GPU (optional) for model training and testing.
* **Software:**
  + **Programming Language:** Python 3.x for developing the chatbot.
  + **Development Environment:** VS Code or PyCharm for coding and debugging.
  + **NLP Libraries:**
    - **NLTK:** For text preprocessing tasks such as tokenization, stopword removal, and lemmatization.
    - **SpaCy:** For advanced NLP tasks like entity recognition and dependency parsing.
    - **Rasa:** For building the chatbot's core functionality, including intent recognition, dialogue management, and entity extraction.
    - **Scikit-learn:** For implementing machine learning models and text vectorization techniques like TF-IDF.
  + **Frameworks and Tools:**
    - **TensorFlow/PyTorch:** For training and deploying custom NLP models if required.
    - **Dialogflow or IBM Watson:** As an alternative for implementing complex chatbot functionalities.
    - **Docker:** For containerizing the chatbot application to ensure consistent deployment across different environments.
    - **Database:** MySQL or MongoDB for storing conversation data and managing user interactions.
    - **Version Control:** Git for tracking changes and collaboration.

**2. Methods**

* **Data Collection:**
  + **Source Identification:** Collect relevant customer service data, including chat logs, FAQs, and email interactions, from various sources such as customer support platforms, company databases, and publicly available datasets.
  + **Data Acquisition:** Use web scraping tools or APIs to gather additional textual data if needed.
* **Data Preprocessing:**
  + **Tokenization:** Split the text data into individual words or tokens using NLTK or SpaCy.
  + **Stopword Removal:** Eliminate common words (e.g., "and," "the") that do not contribute to intent recognition using NLTK’s stopword list.
  + **Lemmatization:** Reduce words to their base or root form (e.g., "running" to "run") to ensure consistency in text analysis.
  + **Entity Recognition:** Identify and categorize key entities (e.g., product names, dates) in the text using SpaCy’s named entity recognition (NER) capabilities.
* **Model Development:**
  + **Intent Recognition:** Train a machine learning model using Rasa or another NLP framework to classify user inputs into predefined intents (e.g., "order status," "return policy").
  + **Entity Extraction:** Implement entity extraction using pre-trained models in Rasa or SpaCy to identify relevant information within the user’s input.
  + **Context Management:** Develop a context-aware dialogue management system using Rasa, allowing the chatbot to handle multi-turn conversations by maintaining the context of the interaction.
* **Chatbot Integration:**
  + **Knowledge Base Integration:** Link the chatbot to a domain-specific knowledge base, enabling it to provide accurate and relevant responses to user queries.
  + **Escalation Mechanism:** Implement a mechanism to route complex or unhandled queries to human agents, ensuring a seamless transition between automated and human support.
* **Testing and Evaluation:**
  + **Unit Testing:** Perform unit tests on individual components of the chatbot, such as intent recognition and entity extraction modules.
  + **Integration Testing:** Ensure that all chatbot components work together smoothly, from data preprocessing to response generation.
  + **User Acceptance Testing:** Conduct user acceptance testing with real users to evaluate the chatbot’s performance in a live environment.
  + **Performance Metrics:** Measure the chatbot’s performance using metrics such as accuracy, precision, recall, and user satisfaction ratings.
* **Deployment:**
  + **Containerization:** Use Docker to containerize the chatbot application, ensuring consistent deployment across different environments.
  + **Deployment:** Deploy the chatbot on a cloud platform such as AWS or Google Cloud, making it accessible to users through web or mobile interfaces.
  + **Monitoring and Feedback:** Monitor the chatbot’s performance post-deployment and gather user feedback to continuously improve the system.

# PYTHON CODE:

# # Import necessary libraries

# import spacy

# import nltk

# from nltk.corpus import stopwords

# from nltk.tokenize import word\_tokenize

# from nltk.stem import WordNetLemmatizer

# from rasa\_sdk import Action, Tracker

# from rasa\_sdk.executor import CollectingDispatcher

# # Download NLTK resources

# nltk.download('punkt')

# nltk.download('stopwords')

# nltk.download('wordnet')

# # Initialize SpaCy model and NLTK components

# nlp = spacy.load('en\_core\_web\_md')

# lemmatizer = WordNetLemmatizer()

# stop\_words = set(stopwords.words('english'))

# # Text preprocessing function

# def preprocess\_text(text):

# # Tokenization

# tokens = word\_tokenize(text.lower())

# 

# # Remove stopwords

# tokens = [word for word in tokens if word.isalpha() and word not in stop\_words]

# 

# # Lemmatization

# tokens = [lemmatizer.lemmatize(word) for word in tokens]

# 

# return tokens

# # Example usage of preprocessing function

# text = "What is the status of my order? I placed it yesterday."

# processed\_text = preprocess\_text(text)

# print(f"Processed Text: {processed\_text}")

# # Rasa Custom Action

# class ActionCheckOrderStatus(Action):

# def name(self) -> str:

# return "action\_check\_order\_status"

# def run(self, dispatcher: CollectingDispatcher,

# tracker: Tracker,

# domain: Dict[str, Any]) -> List[Dict[str, Any]]:

# order\_id = tracker.get\_slot("order\_id")

# # Implement your logic to check order status

# # Example response

# status = "Your order is being processed and will be shipped soon."

# dispatcher.utter\_message(text=f"The status for order {order\_id} is: {status}")

# return []

# # Create configuration files for Rasa

# def create\_rasa\_files():

# with open("nlu.yml", "w") as f:

# f.write("""version: "2.0"

# nlu:

# - intent: greet

# examples: |

# - hello

# - hi

# - hey

# - good morning

# - good evening

# - intent: ask\_order\_status

# examples: |

# - What's the status of my order?

# - Can you check my order status?

# - I want to know if my order has shipped.

# - Where is my order?

# - intent: goodbye

# examples: |

# - bye

# - goodbye

# - see you later

# - talk to you later

# """)

# with open("domain.yml", "w") as f:

# f.write("""version: "2.0"

# intents:

# - greet

# - ask\_order\_status

# - goodbye

# entities:

# - order\_id

# slots:

# order\_id:

# type: text

# responses:

# utter\_greet:

# - text: "Hello! How can I assist you today?"

# 

# utter\_ask\_order\_status:

# - text: "Please provide your order ID."

# 

# utter\_goodbye:

# - text: "Goodbye! Have a great day!"

# session\_config:

# session\_expiration\_time: 60

# carry\_over\_slots\_to\_new\_session: true

# """)

# with open("stories.yml", "w") as f:

# f.write("""version: "2.0"

# stories:

# - story: order status inquiry

# steps:

# - intent: greet

# - action: utter\_greet

# - intent: ask\_order\_status

# - action: utter\_ask\_order\_status

# - intent: provide\_order\_id

# - action: action\_check\_order\_status

# - story: goodbye

# steps:

# - intent: goodbye

# - action: utter\_goodbye

# """)

# with open("actions.py", "w") as f:

# f.write("""from typing import Any, Text, Dict, List

# from rasa\_sdk import Action, Tracker

# from rasa\_sdk.executor import CollectingDispatcher

# class ActionCheckOrderStatus(Action):

# def name(self) -> Text:

# return "action\_check\_order\_status"

# def run(self, dispatcher: CollectingDispatcher,

# tracker: Tracker,

# domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:

# order\_id = tracker.get\_slot("order\_id")

# # Implement your logic to check order status

# # Example response

# status = "Your order is being processed and will be shipped soon."

# dispatcher.utter\_message(text=f"The status for order {order\_id} is: {status}")

# return []

# """)

# # Run the script to create Rasa configuration files

# create\_rasa\_files()

# print("Rasa configuration files created successfully.")

# CONCLUSION

In conclusion, the development of a context-aware chatbot for customer support illustrates the significant impact of utilizing advanced NLP techniques and frameworks like Rasa to enhance interaction quality and response accuracy. By implementing comprehensive text preprocessing, effective context management, and tailored custom actions, the chatbot is capable of addressing a diverse range of customer queries with increased relevance and coherence. The rigorous testing and refinement processes ensure that the chatbot performs reliably and meets user expectations, while feedback from real-world use provides valuable insights for continuous improvement. Future advancements, including the expansion of intents and entities, integration of more sophisticated NLP models, and seamless connection with additional systems, promise to further enhance the chatbot’s functionality and effectiveness. Overall, this project highlights the transformative potential of conversational AI in improving customer support and delivering a more personalized and efficient user experience.

# REFERENCES

1. Adamopoulou, E., & Moussiades, L. (2020). An overview of chatbot technology. In *IFIP international conference on artificial intelligence applications and innovations* (pp. 373-383). Springer, Cham.
2. Rahman, A. M., Al Mamun, A., & Islam, A. (2017, December). Programming challenges of chatbot: Current and future prospective. In *2017 IEEE region 10 humanitarian technology conference (R10-HTC)* (pp. 75-78). IEEE.
3. Zhou, L., Gao, J., Li, D., & Shum, H. Y. (2020). The design and implementation of xiaoice, an empathetic social chatbot. *Computational Linguistics*, *46*(1), 53-93.
4. AbuShawar, B., & Atwell, E. (2015). ALICE chatbot: Trials and outputs. *Computación y sistemas*, *19*(4), 625-632.
5. Hiremath, Guruswami, Aishwarya Hajare, Priyanka Bhosale, Rasika Nanaware, and K. S. Wagh. "Chatbot for education system." *International Journal of Advance Research, Ideas and Innovations in Technology* 4, no. 3 (2018): 37-43.
6. Xu, A., Liu, Z., Guo, Y., Sinha, V., & Akkiraju, R. (2017, May). A new chatbot for customer service on social media. In *Proceedings of the 2017 CHI conference on human factors in computing systems* (pp. 3506-3510).
7. Abdul-Kader, S. A., & Woods, J. C. (2015). Survey on chatbot design techniques in speech conversation systems. *International Journal of Advanced Computer Science and Applications*, *6*(7).
8. Sharma, V., Goyal, M., & Malik, D. (2017). An intelligent behaviour shown by chatbot system. *International Journal of New Technology and Research*, *3*(4), 263312.
9. Chaves, A. P., & Gerosa, M. A. (2021). How should my chatbot interact? A survey on social characteristics in human–chatbot interaction design. *International Journal of Human–Computer Interaction*, *37*(8), 729-758.
10. Al-Zubaide, H., & Issa, A. A. (2011, November). Ontbot: Ontology based chatbot. In *International Symposium on Innovations in Information and Communications Technology* (pp. 7-12). IEEE.
11. Trivedi, A., Gor, V., & Thakkar, Z. (2019). Chatbot generation and integration: A review. *International Journal of Advance Research, Ideas and Innovations in Technology*, *5*(2), 1308-1311.
12. Nagarhalli, T. P., Vaze, V., & Rana, N. K. (2020, March). A review of current trends in the development of chatbot systems. In *2020 6th International conference on advanced computing and communication systems (ICACCS)* (pp. 706-710). IEEE.
13. Csaky, R. (2019). Deep learning based chatbot models. *arXiv preprint arXiv:1908.08835*.
14. Jenkins, M. C., Churchill, R., Cox, S., & Smith, D. (2007). Analysis of user interaction with service oriented chatbot systems. In *Human-Computer Interaction. HCI Intelligent Multimodal Interaction Environments: 12th International Conference, HCI International 2007, Beijing, China, July 22-27, 2007, Proceedings, Part III 12* (pp. 76-83). Springer Berlin Heidelberg.
15. Skjuve, M., Følstad, A., Fostervold, K. I., & Brandtzaeg, P. B. (2021). My chatbot companion-a study of human-chatbot relationships. *International Journal of Human-Computer Studies*, *149*, 102601.