CHATBOT WITH EFFECTIVE CONVERSATIONS

CASE STUDY: E-COMMERCE CHATBOT ASSISTANT

PROJECT FINAL

DECEMBER 18, 2023

GROUP 3

- **Supervisor: Prof. Dr. Amdad Khan**
- **Team members:**
  - *616917: Geoffrey Duncan Opiyo*
  - *616965: Phuong Khanh Nguyen*
  - *616947: Deo Mugabe*
  - *616922: Duc Phi Ngo*

Table of Contents

Abstract

Chatbots with AI are changing the way people shop. AI and technologies that help computers understand human language (NLP) will likely speed up this change even more. But customers usually prefer talking to real people instead of chatbots, which often seem cold and not very human-like. Even though there's a push to make chatbots seem more human, we don't know much about how giving them human-like ways of speaking affects how personalized products feel or if people are willing to pay more for products in situations where they talk to chatbots [1].

In our project, we created a chatbot using a Large Language Model and a Learning Agent. We discovered that this chatbot can interact with customers on our retail website to recommend products they might like. This project lays the groundwork for future AI-powered chatbots designed to offer customized and data-informed product suggestions.

We plan to enhance the project by incorporating a knowledge graph and using MySQL for product data storage. Additionally, we aim to give the chatbot a human-like voice for voice-based customer interactions.

Keywords: AI, Chatbots, Conversational commerce, Learning Agent, Reinforcement Learning, LLM, LLM embedding.

1. Problem statement and project introduction

In the modern digital marketplace, customer engagement is paramount. However, the prevalent use of traditional chatbots, often lacking in personalization and intelligence, has led to a significant gap in consumer satisfaction. This paper proposes an innovative solution: a chatbot integrated with advanced natural language processing (NLP) and intelligent agent capabilities. This hybrid system aims to deliver a personalized shopping experience, resembling human-like interactions while providing efficient and accurate customer support.

Consumers have shown a consistent preference for human interaction, often finding traditional chatbots impersonal and detached, which hinders the overall customer experience. To bridge this gap, we propose a chatbot enhanced with natural language processing and intelligent agent technology, designed to emulate the nuanced communication of human interaction. This advanced chatbot serves as a personal shopping assistant, intuitively offering product recommendations aligned with individual customer preferences and utilizing a Large Language Model to elevate customer support by answering queries, providing comprehensive product details, and facilitating informed decision-making.

2. Agent and its architecture

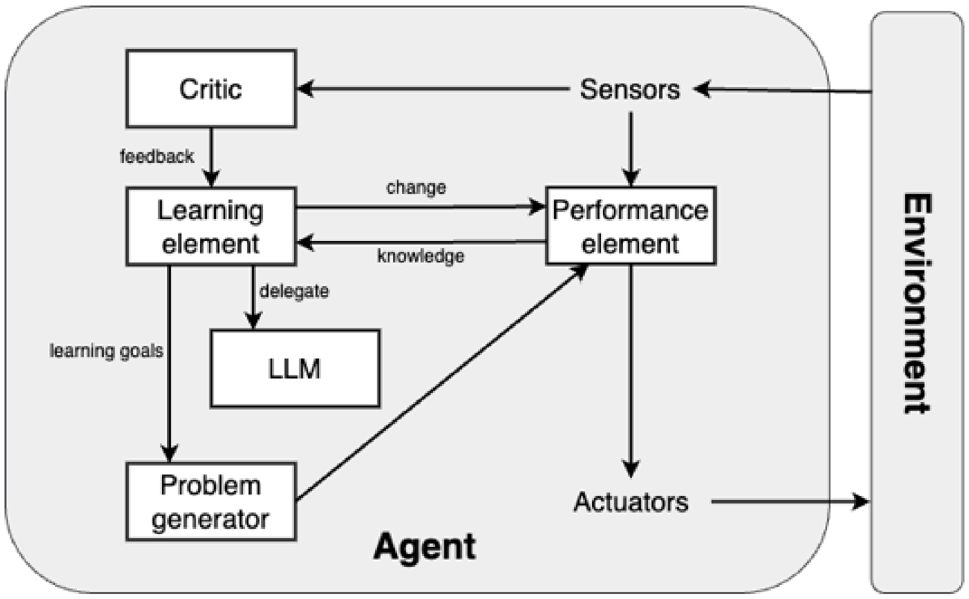


Figure 1: System architecture of the sales bot

Learning Agent: This entity is at the forefront of adaptability within the architecture. It's designed to evolve by learning from interactions and feedback. It includes:

Critic: The critic's role is akin to a rigorous peer-review process. It evaluates the actions taken by the performance element against a set of criteria or performance standards, providing essential feedback for improvement.

Learning Element: Utilizing the feedback from the critic, the learning element updates the agent's knowledge base.

Problem Generator: This proactive component suggests challenges to solve, encouraging the exploration of new strategies or knowledge. It's like a researcher who devises new experiments to probe the unknowns of the universe.

Performance Element: The 'executor' of the agent, it uses the knowledge to decide on actions, commanding the actuators to interact with the environment. You can compare this to the autopilot of an aircraft, making real-time decisions based on sensor data.

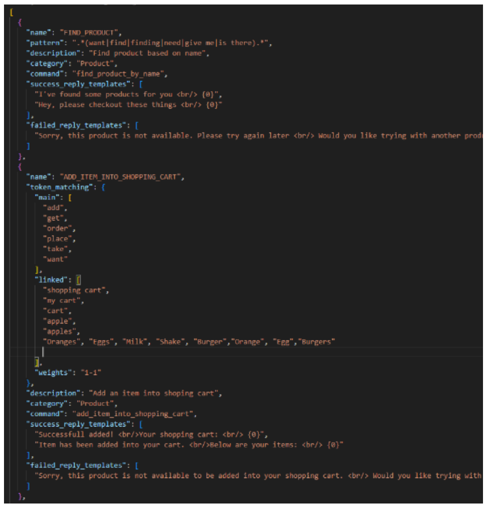
Large Language Model (LLM): Here, the LLM is part of the Learning Agent, where it serves as an advanced component capable of understanding and generating human-like text. It uses vast amounts of data and complex algorithms to interpret input, generate responses, solve problems, and even simulate conversation. In this architecture, the LLM could be delegated certain tasks by the learning element, such as processing natural language data or generating hypotheses.

Sensors and Actuators: These are the physical or virtual apparatus that the agent uses to perceive its environment (sensors) and to take action within it (actuators). In the realm of AI, sensors could be data inputs, and actuators could be any kind of output the system is capable of, including generating text or making changes to a database.

Environment: This is the domain in which the agent operates. For a Large Language Model, the environment could be the datasets it was trained on, the input it receives during interactions, or the broader context it's applied to, such as a dialogue system or a content creation platform.

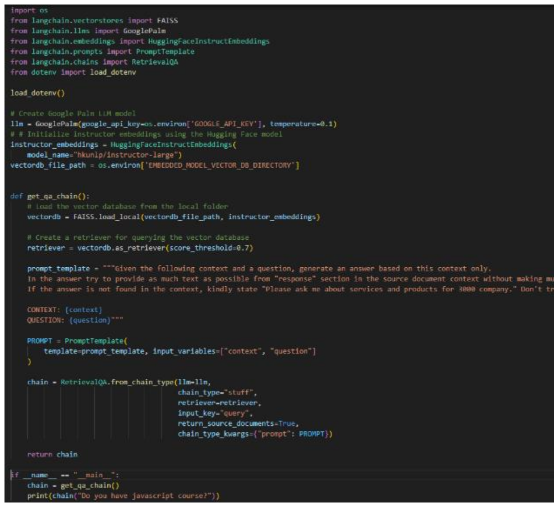
The information and control flows indicated by the arrows show that the sensors inform the performance element about the state of the environment. The learning element adjusts the knowledge base, potentially influencing future actions of the performance element. The critic assesses the suitability of these actions in achieving the agent's goals, and the problem generator introduces new challenges to foster learning.

In essence, this architecture with an LLM at its core describes a sophisticated, learning-oriented AI system capable of interacting with and adapting to its environment through a continuous feedback loop.



```
[
  {
    "name": "FIND_PRODUCT",
    "pattern": ".*(want|find|finding|need|give me|is there).*",
    "description": "Find product based on name",
    "category": "Product",
    "command": "find_product_by_name",
    "success_reply_templates": [
      "I've found some products for you <br/> {0}",
      "Hey, please checkout these things <br/> {0}"
    ],
    "failed_reply_templates": [
      "Sorry, this product is not available. Please try again later <br/> Would you like trying with another prod
    ]
  },
  {
    "name": "ADD_ITEM_INTO_SHOPPING_CART",
    "token_matching": {
      "main": [
        "add",
        "get",
        "order",
        "place",
        "take",
        "want"
      ],
      "linked": [
        "shopping cart",
        "my cart",
        "cart",
        "apple",
        "apples",
        "Oranges", "Eggs", "Milk", "Shake", "Burger","Orange", "Egg","Burgers"
        
      ],
      "weights": "1-1"
    },
    "description": "Add an item into shoping cart",
    "category": "Product",
    "command": "add_item_into_shopping_cart",
    "success_reply_templates": [
      "Successfull added! <br/>Your shopping cart: <br/> {0}",
      "Item has been added into your cart. <br/>Below are your items: <br/> {0}"
    ],
    "failed_reply_templates": [
      "Sorry, this product is not available to be added into your shopping cart. <br/> Would you like trying with
    ]
  },
```

Figure 2: Deductive Reasoning (JSON response)



```
import os
from langchain.vectorstores import FAISS
from langchain.llms import GooglePalm
from langchain.embeddings import HuggingFaceInstructEmbeddings
from langchain.prompts import PromptTemplate
from langchain.chains import RetrievalQA
from dotenv import load_dotenv

load_dotenv()

# Create Google Palm LLM model
llm = GooglePalm(google_api_key=os.environ['GOOGLE_API_KEY'], temperature=0.1)
# # Initialize instructor embeddings using the Hugging Face model
instructor_embeddings = HuggingFaceInstructEmbeddings(
    model_name="hkunlp/instructor-large")
vectordb_file_path = os.environ['EMBEDDED_MODEL_VECTOR_DB_DIRECTORY']


def get_qa_chain():
    # Load the vector database from the local folder
    vectordb = FAISS.load_local(vectordb_file_path, instructor_embeddings)

    # Create a retriever for querying the vector database
    retriever = vectordb.as_retriever(score_threshold=0.7)

    prompt_template = """Given the following context and a question, generate an answer based on this context only.
    In the answer try to provide as much text as possible from "response" section in the source document context without making m
    If the answer is not found in the context, kindly state "Please ask me about services and products for 3000 company." Don't tr

    CONTEXT: {context}
    QUESTION: {question}"""

    PROMPT = PromptTemplate(
        template=prompt_template, input_variables=["context", "question"]
    )

    chain = RetrievalQA.from_chain_type(llm=llm,
                                        chain_type="stuff",
                                        retriever=retriever,
                                        input_key="query",
                                        return_source_documents=True,
                                        chain_type_kwargs={"prompt": PROMPT})

    return chain

if __name__ == "__main__":
    chain = get_qa_chain()
    print(chain("Do you have javascript course?"))
```

Figure 3: LLM Model (Inductive Reasoning)

The intelligent agent uses these predefined rules to parse user input and determine the correct action for its actuators, in Figure 2, actions like searching for a product or adding or removing items to shopping cart. The rules are designed to understand the intent behind the user's input and to provide appropriate responses or actions based on that input. This is a fundamental part of how our chatbot simulates understanding and provides interactive responses.

Furthermore, Figure 3 shows a Python script designed to implement an LLM (Large Language Model) using Google's PaLM for answering questions. The code is structured to use prompt engineering to guide the LLM in providing responses that are contextually relevant and constrained to the information it has been trained on or has access to.

3. Embeddings overview

We need to pull out the text from the excel document containing the retail information about the retail shop and break it into smaller sections. This step is crucial because there's a maximum amount of text - called tokens - that a Large Language Model (LLM) can handle at once. By dividing the text into sections, we can feed each part to the LLM separately, ensuring no details are missed and the LLM can consider the whole document. Next, we turn these text sections into a form the LLM can understand better, known as embeddings. We saved them in Faiss (Figure 3) the vector database. After this, when a user asks a question, we transform their query into embeddings with the LLM. We then search for the text sections most likely to contain the answer by comparing how similar their embeddings are to the query's embeddings. We present the most relevant sections to the LLM along with the question, asking it to come up with an answer. We do this with each relevant section because sometimes a single topic might be split across several sections during the breakup of the text, and we want to consider all related information to provide the best answer [2]

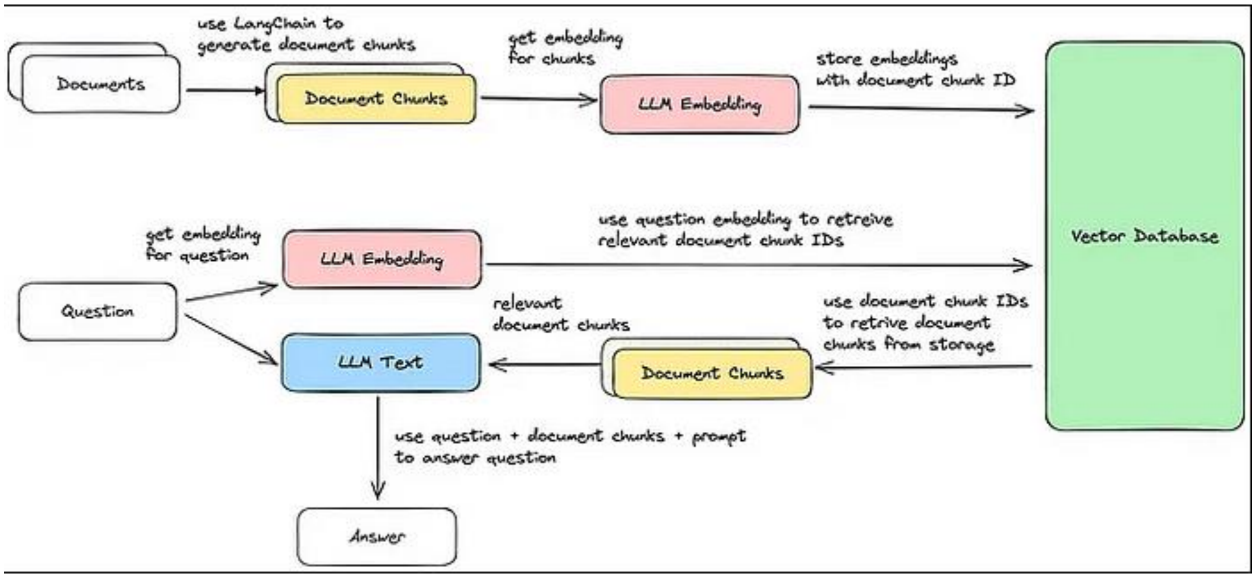
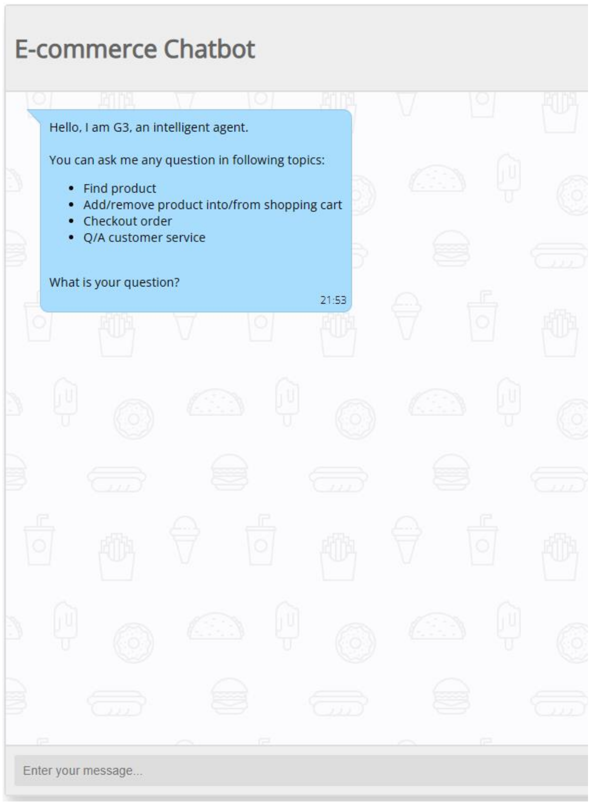

use LangChain to generate document chunks
get embedding for chunks
store embeddings with document chunk ID
Documents
Document Chunks
LLM Embedding
Vector Database
get embedding for question
use question embedding to retreive relevant document chunk IDs
LLM Embedding
Question
relevant document chunks
use document chunk IDs to retrive document chunks from storage
LLM Text
Document Chunks
use question + document chunks + prompt to answer question
Answer

Figure 4: Architecture diagram showing the flow of the solution [1].

Finally, the fully developed question-answering system is deployed, ready to cater to a wide array of user queries and needs.

4. Study cases.

Study 1: In this study, we ask chatbot show us what they have in the store, for example, the chatbot give us some recommendations about apples, and display apple items on the chat.



Hello, I am G3, an intelligent agent.

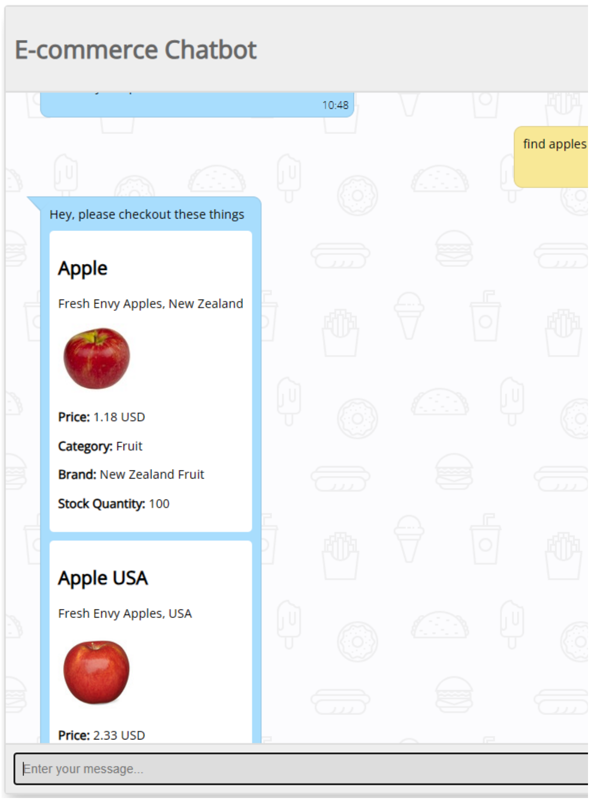
You can ask me any question in following topics:

- Find product
- Add/remove product into/from shopping cart
- Checkout order
- Q/A customer service

What is your question?

21:53

Figure 5: GUI of chatbot with 4 use-cases



find apples

Hey, please checkout these things

Apple

Fresh Envy Apples, New Zealand

Price: 1.18 USD

Category: Fruit

Brand: New Zealand Fruit

Stock Quantity: 100

Apple USA

Fresh Envy Apples, USA

Price: 2.33 USD

Figure 6: Chatbot answering the customer query

Figure 5 shows the interface of e-commerce chatbot and Figure 6 displays a conversation with an e-commerce chatbot interface where a customer has inquired about apples. Here's what the chatbot's response indicates:

The chatbot has understood the query for apples and has provided two different product options. Indeed, the first product is labeled "Apple," specifically "Fresh Envy Apples, New Zealand." It is priced at 1.18 USD, categorized under "Fruit," with the brand "New Zealand Fruit," and there is a stock quantity of 100 available for purchase. The second product is "Apple USA," also "Fresh Envy Apples," but sourced from the USA.

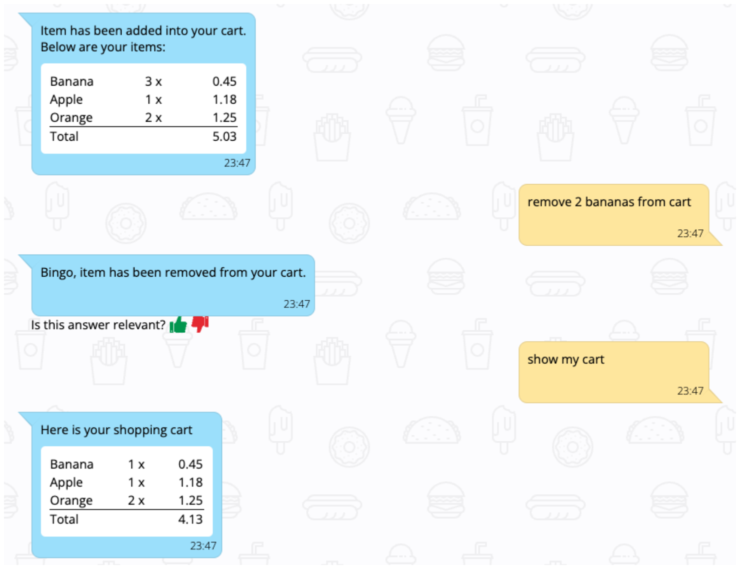
This option is priced higher at 2.33 USD. Each product option comes with an image of the apple, which helps customers to visually identify the product. The chatbot's prompt "Hey, please check out these things" suggests a casual, friendly tone aimed at encouraging the customer to consider the options provided.

Overall, the chatbot has efficiently processed the customer's request by pulling relevant product details from the e-commerce platform’s database, illustrating a simple yet effective use of AI in a retail setting to assist customers.

Study 2: This study tests to see if our chatbot can learn from user feedback as a deductive learning.

Learn from feedback

```
FEEDBACK_GENERATION_RATE=80
```



Item has been added into your cart.
Below are your items:

Banana	3 x	0.45
Apple	1 x	1.18
Orange	2 x	1.25
Total		5.03

23:47

remove 2 bananas from cart
23:47

Bingo, item has been removed from your cart.
23:47

Is this answer relevant? 👍 👎

show my cart
23:47

Here is your shopping cart

Banana	1 x	0.45
Apple	1 x	1.18
Orange	2 x	1.25
Total		4.13

23:47

Figure 7: The demo of Chatbot removing items.

Figure 7 shows a conversation where the chatbot confirms that items have been added to a shopping cart, detailing quantities and prices for bananas, apples, and oranges, and providing a total cost. In a subsequent interaction, the chatbot acknowledges the removal of an item from the cart after the user's command to "remove 2 bananas from cart."

To the left, under the header "Learn from feedback," we see "FEEDBACK_GENERATION_RATE=80" which suggests a numerical value associated with the chatbot's learning process. This represents a parameter in a reinforcement learning algorithm, where a higher feedback generation rate indicates more frequent feedback to the chatbot. In the context of reinforcement learning, this rate could potentially influence how often the chatbot updates its responses or actions based on user interactions or feedback, with the aim of improving its performance over time. Furthermore, chatbot responses to a prompt asking if the provided answer is relevant, to which a thumbs-up reaction is displayed, likely indicating positive feedback.

Finally, the chatbot displays the updated shopping cart after the changes, showing the reduced quantities of bananas and the new total cost.

This simulated chat flow gives an example of how a chatbot can interact with users in an online shopping environment, executing commands to add or remove items from a cart, and potentially using feedback to learn and adjust its actions, enhancing the user experience through a process akin to reinforcement learning.

5. Conclusion and future work.

In this study, we presented an intelligent chatbot tailored for e-commerce platforms, capable of responding to human queries both automatically and with a level of intelligence that mimics human reasoning. Utilizing a blend of deductive reasoning and inductive learning, the chatbot adapts through customer feedback and crafts responses that are informed by a Large Language Model. Drawing inspiration from the principles of Reinforcement Learning, it accumulates rewards to refine its decision-making process.

While promising, the project is not without its limitations. The chatbot's development was constrained by time, which may have impacted the depth of learning and the range of features currently available. Looking ahead, we plan to expand the chatbot's capabilities by incorporating a knowledge graph to enhance query understanding. Additionally, we aim to develop voice command features to provide a more accessible and seamless user experience. As we continue to refine the chatbot's functionalities, we will explore the integration of multimodal feedback processing to understand and interpret non-textual customer inputs, such as emotions or images, which will further personalize and humanize the shopping experience. Ultimately, our goal is to create a chatbot that not only understands the customer's needs but anticipates them, paving the way for a new era of digital customer service.

References

[1] S. Heyer, "Generative AI - Document Retrieval and Question Answering with LLMs," 4 June 2023. [Online]. Available: https://medium.com/google-cloud/generative-ai-document-retrieval-and-question-answering-with-llms-2b0fb80ae76d.

[2] Chen, Huajun et al., "Neural symbolic reasoning with knowledge graphs: Knowledge extraction, relational reasoning, and inconsistency checking," *Fundamental Research,* 2021.

[3] Mohammad Monirujjaman Khan, Shahnoor Chowdhury Eshan, "Development of An e-commerce Sales Chatbot," 2021.