A Major Project Final Report on

AI-based Pluggable Chatbot

Submitted in Partial Fulfillment of the Requirements for

The Degree of Bachelor of Engineering in Information Technology

Under Pokhara University

Submitted by:

Dishan Shrestha (181409)

Rinku Deuja (181450)

Sadika Kasaju (181432)

Simran Tamrakar (181444)

Under the supervision of:
Mr. Bhusan Sumsher Thapa

Date: **13-July-2023**

Department of IT Engineering

Nepal College of Information Technology

Affiliated to Pokhara University

ACKNOWLEDGEMENT

We are glad to present our final project report which is a part of the Engineering Curriculum under Pokhara University. This project would not be possible without the guidance of our seniors and teachers. It is an utmost privilege to express our sincere regards to our project supervisor, **Mr. Bhusan Sumsher Thapa** for his valuable guidance, encouragement, and support,
throughout the duration of the project.

We take this opportunity to thank our professor, **Dr. Roshan Chitrakar** for providing us the opportunity to work on this project. We would also like to thank our teachers and department head for their efforts and valuable time throughout the project development that led to its completion.

Finally, we would also like to thank all our friends that have contributed to our project with their suggestions and critiques.

Abstract

This project is aimed at creating a chatbot that simulates human conversation through text. They can be integrated into websites to assist users by providing recommendations, answering queries and other assistance needs. This chatbot will be able to respond to user queries with relevant information and make meaningful conversation. Our chatbot will be able to switch between closed and open domain depending upon the requirement using various tools and techniques. The primary use of this chatbot is to automate customer service interactions and improve user experience. It uses Natural Language Processing (NLP) and machine learning algorithms to understand the user's input and generate a relevant response. For NLP, we will be using Bidirectional Encoder Representations from Transformer (BERT) along with a neural network enabling the creation of more advanced and intelligent systems that are capable of understanding and responding to natural language inputs in a human-like way. We have used haystack with elastic search as document database for the chatbot to serve conversational needs. This project is developed by directly connecting every step of the development process to a corresponding testing activity using verification and validation model.

Keywords: Python, NLP, TensorFlow, Elastic Search, Haystack, BERT, Pandas

Contents

LIST OF FIGURES

LIST OF TABLES

1. Introduction

Our Project “AI-based Pluggable Chatbot” will generate automated and conversational responses to user inquiries. AI-based chatbots have revolutionized the way businesses interact with their customers. These intelligent virtual assistants use artificial intelligence and natural language processing to communicate with users and provide them with personalized assistance. Our pluggable chatbot can be integrated into different websites and help organizations facilitate their users by providing assistance through our chatbot.

Our chatbot has used two different types of models: one is a closed domain chatbot and the other is an open domain system. Our BERT based conversational chatbot can understand the user's intent and provide relevant responses by analyzing the input text. We have also used elastic search as backend database and BM25 algorithm to switch the chatbot and make it conversational. It can also handle complex questions and provide accurate answers, making it an indispensable tool for businesses looking to improve their customer service. Our chatbots can be incorporated into applications that involve communication and interaction with users, and can be customized to meet the specific needs and requirements of the application. By training the chatbot’s NLP model on a dataset of sample conversations, our chatbot is able to aid in customer service, engagement, and support by replacing human support agents with AI and automation technologies that can communicate with end-users via chat.

1.1. Problem Statement

Some common problems that an AI-based chatbot aims to address include:

- Long response times: Many companies have trouble answering customers' questions promptly, which may result in dissatisfied customers.
- High cost of customer service: It may be expensive, especially for small organizations, to hire and train human customer service personnel.
- Poor service quality: Human interaction can be influenced by emotions, leading to a bad user experience.
- Limited availability: Customers who want assistance outside of such hours may find it inconvenient since human customer service professionals are only available during particular hours.

- Lack of personalization: Based on client behavior and preferences, traditional customer care channels like email and phone assistance might not be able to offer individualized advice or replies.

1.2. Project Objectives

The main aim of creating "AI-based chatbot" is to improve efficiency, productivity, and customer satisfaction by providing instant and personalized assistance, without requiring human intervention. By automating routine tasks and inquiries, chatbots can save time and resources, and enable organizations to focus on higher value activities.

1. To implement BERT model for intent based and extractive chatbot
2. To provide pluggable chatbot widget that can be integrated in all kinds of web applications.

1.3. Significance of the study

The significance of our study is:

- Provide 24/7 customer support: It ensures that customers can receive information or assistance at any time. It also enables business to grow and expand to a larger level.
- Allow users to engage in both open and closed domain Q&A: It allows users to throw queries based on the specific domain as well as to ask questions from any domain or topic. It enhances the versatility of the chatbot as well as the user experiences.
- Works on both labeled and unlabeled text corpus: This allows the chatbot to handle predefined categories and provide accurate response based on the categorized data. Also the chatbot is capable of adapting to new or evolving user needs by processing unlabeled data and extracting relevant information from it.
- Instant Response: It allows the user to save their precious time as the chatbot is able to respond to the users from its both closed and open Q&A model. Not only it provides users with immediate response but also with accuracy most of the time.
- Reduce confusion leading from lexical and grammatical errors: It ensures that the chatbot accurately understands the user's query providing clear and coherent responses. By overcoming language related challenges, the chatbot improves the overall effectiveness of communication, enhancing user experience and minimizing potential misunderstandings.

- Increase customer engagement and conversion rate: By providing personalized recommendations, timely assistance and relevant information, the chatbot boosts customer engagement, which in turn increases the likelihood of successful conversions and repeat business.

By providing prompt and correct answers to consumer inquiries, this project enables businesses to provide better customer service. As a result of our effort, chatbots that can communicate without human intervention will be created.

1.4. Scope and Limitation

The scope of this project is to serve users with web applications capable of automating chat conversations. This project includes following scopes:

- Understand user queries and provide appropriate responses.
- Automatic information extraction.
- Set predefined answers as per the business requirement.
- Provide the option of switching between models that best support the user requirements.

It also comes with few limitations:

- Supports only the English language.
- Only supports text.
- Accuracy depends upon text corpus for extractive chatbot.
- Some responses might be vague and error prone.

2. Literature Review

2.1. ChatGpt:

ChatGPT, developed by OpenAI, is an advanced language model based on the GPT (Generative Pre-trained Transformer) architecture. It has gained significant attention due to its remarkable ability to generate contextually relevant and coherent responses in conversational settings. In this article, we will explore the construction and development of ChatGPT, as well as its key features that make it a powerful tool in natural language processing.

Building ChatGPT:

ChatGPT is built upon the Transformer architecture, which has revolutionized the field of NLP. The Transformer model utilizes self-attention mechanisms to capture contextual relationships between words, allowing it to understand and generate coherent sentences. The training process for ChatGPT involves pre-training and fine-tuning stages.

During pre-training, ChatGPT is exposed to a large corpus of text data from the internet, enabling it to learn the statistical patterns and relationships within language. The model learns to predict the next word in a sentence based on the context provided by the preceding words. This process helps ChatGPT develop a strong foundation of language understanding.

In the fine-tuning stage, the model is further trained on specific datasets with human-generated conversations. The data consists of dialogue exchanges where the model learns to generate responses that are contextually appropriate. This fine-tuning process enables ChatGPT to produce more accurate and coherent replies during interactive conversations.

Development of ChatGPT:

OpenAI has released multiple versions of ChatGPT, each iteration refining the model's capabilities and addressing its limitations. The initial versions, such as GPT and GPT-2, showcased impressive language generation but suffered from a lack of control and sensitivity to input instructions. As a result, these models were prone to generating biased or inappropriate responses.

To mitigate these issues, OpenAI introduced the concept of "prompts" in the development of ChatGPT.[10] By providing users with the ability to input a prompt that sets the behavior and context for the conversation, OpenAI aimed to make the model more controllable and aligned with

user preferences. Prompt engineering techniques, such as providing explicit instructions or adding user-specific personal information, have been employed to improve the relevance and coherence of generated responses.

Features of ChatGPT:

- Contextual Understanding: ChatGPT excels in understanding the context of a conversation. It captures the nuances and dependencies between words and sentences, enabling it to generate responses that are coherent and relevant to the ongoing dialogue.

- Coherent Response Generation: ChatGPT is capable of producing human-like responses that are grammatically correct and contextually appropriate. Its ability to generate coherent and fluent sentences enhances the user experience and makes conversations with the chatbot feel more natural.

- Flexibility and Adaptability: ChatGPT is designed to handle a wide range of conversational topics and adapt to different user inputs. It can provide information, answer questions, engage in small talk, or assist with specific tasks, making it a versatile conversational agent.

- Interactive Conversations: ChatGPT supports interactive conversations, allowing users to have back-and-forth exchanges. The model can maintain context over multiple turns and provide responses that align with the evolving dialogue, making it suitable for chatbot applications.

- Prompt Engineering: With the introduction of prompts, ChatGPT offers users more control over the generated responses. By providing explicit instructions or specific contexts, users can guide the model's behavior and achieve desired outcomes, enhancing the usability and customization of the chatbot

ChatGPT, built upon the powerful Transformer architecture, has emerged as a prominent language model in the field of conversational AI. Its development has seen iterative improvements in response quality and control, addressing concerns related to bias and sensitivity. With its contextual understanding, coherent response generation, and adaptability, ChatGPT has the

potential to revolutionize chatbot applications across various industries and domains. OpenAI's ongoing efforts to refine and enhance the model's capabilities contribute to the advancement of conversational AI and NLP as a whole.

2.2. Snapchat AI:

Snapchat, the popular social media platform, has integrated various artificial intelligence (AI) technologies to enhance user experiences and offer unique features. In this article, we will explore the building, development, and features of Snapchat AI, which powers the platform's innovative augmented reality (AR) effects, face filters, and content recommendations.

Snapchat has recently started a built-in chatbot feature within its platform. It allows users to ask any type of queries and answers them to the best of its capabilities.

Snapchat has recently started a built-in chatbot feature within its platform. It can answer trivia questions, offer gift advice, help plan trips, and suggest dinner ideas, among other things. However, the responses may sometimes be biased or incorrect. It allows users to ask any type of queries and answers them to the best of its capabilities. Snapchat's My AI chatbot [11] was created using OpenAI's ChatGPT technology, which is a machine learning model that is based on the transformer architecture. Transformers with their attention mechanism, have particularly excelled in generating high-quality responses by capturing contextual information. Snapchat's AI is trained on a vast amount of text data from the internet, which allows it to understand and replicate human language patterns.

OpenAI's ChatGPT technology leverages the power of large-scale deep learning models and transformer architectures to generate text-based responses. Initially, the model undergoes pre-training by learning from a vast corpus of internet text, which helps it acquire grammar, factual knowledge, reasoning abilities and some common sense. Then, during the fine-tuning phase, the model is trained on a more specific dataset with the guidance of human reviewers who follow provided guidelines. This iterative feedback loop refines the model's responses and mitigates biases. When interacting with ChatGPT, users input prompts or messages, which the model processes to generate responses based on its learned patterns. Response generation can involve sampling words probabilistically or ranking from a set of potential responses. Additionally, ChatGPT may sometimes ask for clarifications to enhance the quality of the interaction.

Snapchat has implemented additional safety controls to ensure that My AI always provides positive and uplifting responses to users and avoids generating harmful or negative opinions.

Building Snapchat AI:

Snapchat AI is built on a combination of computer vision, deep learning, and natural language processing techniques. The development of Snapchat AI involves several key components:

- Computer Vision: Computer vision algorithms analyze and interpret visual data from images and videos captured by Snapchat's camera. These algorithms can detect and track facial features, objects, and landmarks, enabling the platform to apply AR effects and filters accurately.
- Deep Learning: Deep learning models, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), are utilized in Snapchat AI to perform various tasks, including image recognition, object detection, and semantic segmentation. These models are trained on vast amounts of labeled data to learn patterns and make accurate predictions.
- Natural Language Processing: Natural language processing techniques enable Snapchat AI to understand and process textual data, such as captions, user comments, and chat messages. Sentiment analysis, entity recognition, and language generation algorithms are employed to extract meaning and provide relevant content recommendations.

Development of Snapchat AI:

Snapchat's AI capabilities have evolved over time, driven by advancements in deep learning and computer vision research. The company invests in research and development to improve the accuracy, speed, and versatility of its AI models. Snapchat also collaborates with academic institutions and industry experts to explore emerging technologies and push the boundaries of AR and AI integration.

One significant milestone in Snapchat AI's development was the acquisition of computer vision startup Looksery in 2015. This acquisition brought advanced facial tracking and AR effects to the platform, laying the foundation for Snapchat's iconic face filters.

Snapchat AI, driven by computer vision, deep learning, and natural language processing techniques, powers a range of innovative features on the platform. From interactive face filters and augmented reality effects to personalized content recommendations, Snapchat AI enhances user experiences and fosters creativity. The continuous development and integration of AI technologies by Snapchat demonstrate the platform's commitment to providing engaging and visually immersive interactions for its users.

3. Methodology

3.1 Software Development Life Cycle

V model (Verification and Validation model)

Verification: It is a process of evaluating the product development phase to ensure that specified requirements are met. This process uses static analysis techniques such as reviews, which are performed without executing the code.

Validation:

It is a process that takes place after the completion of the development phase, which involves dynamic analysis techniques like functional and non-functional testing, performed by executing the code. The objective of the validation process is to evaluate the software and determine if it meets the customer's expectations and requirements.

The V-model is structured around linking each stage of development to a corresponding testing phase, meaning that every step of development is directly connected to a testing activity. Progress to the next phase is only possible once the preceding phase has been completed, so for each development activity, there is a corresponding testing activity.

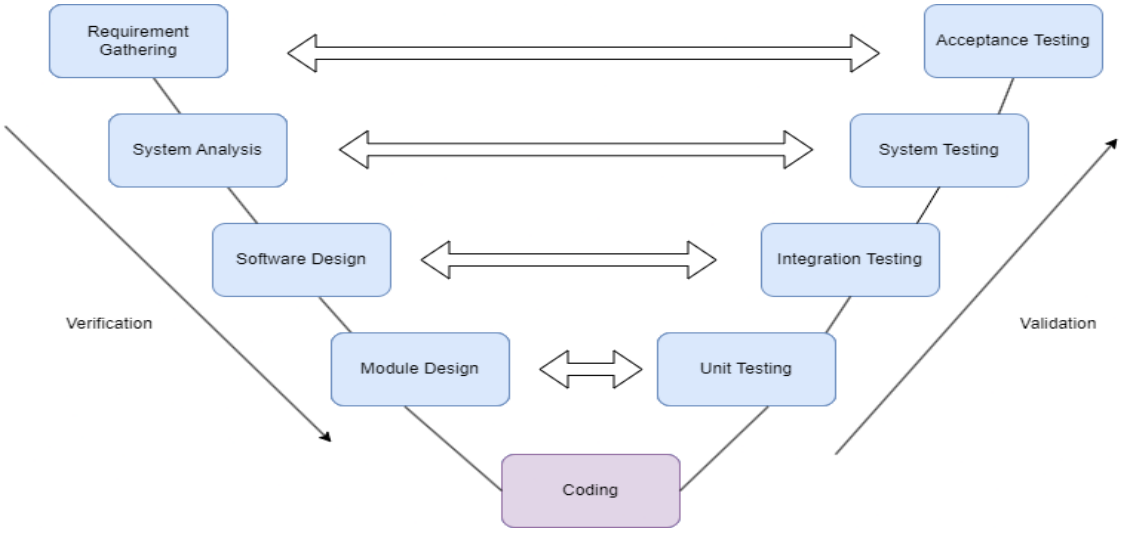

Figure 1: Software Development Life cycle

Design phases:

- Requirement Analysis: This phase contains detailed communication with the customer to comprehend their requirements and expectations. This stage is commonly referred to as Requirement Gathering
- System Design: This phase contains the system design and the complete hardware and communication setup for developing the product.
- Architectural Design: System Design is divided into smaller modules that perform distinct functions. These modules are designed to understand the data transfer and communication between them and with external systems.
- Module Design: During the module design phase, the system is broken down into smaller modules, and a detailed design of each module is specified. This process is also known as Low-Level Design (LLD).

Testing phases:

- Unit Testing: During the module design phase, Unit Test Plans are created. These plans are used to identify and eliminate any errors or bugs in the code at the unit level. Once the plans have been developed, they are executed.
- Integration Testing: Once the unit testing is completed, integration testing is carried out where the modules are integrated, and the system is tested. This test is performed on the architecture design phase and aims to verify the communication of modules among themselves.
- System Testing: System testing involves testing the entire application, including its functionality, inter-dependency, and communication. This type of testing verifies both the functional and non-functional requirements of the developed application.
- User Acceptance Testing: UAT is conducted in an environment that closely resembles the production environment used by the end-user. UAT is performed to ensure that the delivered system meets the user's requirements and that the system is ready to be used in the real world.

3.1.1. Use Case Diagram

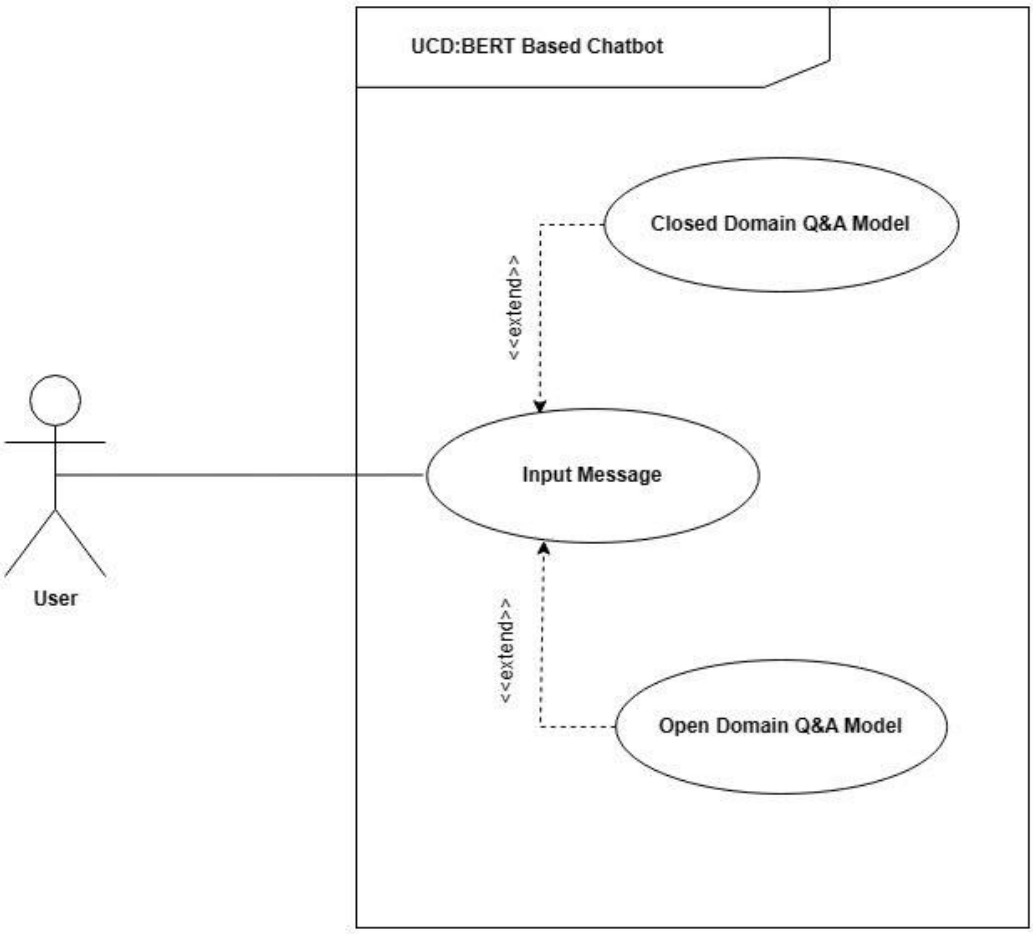

Figure 2: Use Case Diagram

Use Case 1: User Sends Closed Domain Q&A Input Message

Table 1:Use Case 1

Use Case 1	User sends closed domain Q&A input message
Description	The user sends input messages containing specific queries within a predefined domain or topic.
Primary Actors	User, system
Preconditions	The system is running and has a closed domain Q&A model available
Flow of Events	1.The user enters a query as an input message on a topic.
	2.The system receives input messages.
	3.The system uses closed domain Q&A model to process the query
	4.The closed domain Q&A model generates a relevant response based on the query.
Postconditions	The system delivers the response to the user.
Exceptions	The closed domain Q&A model is not available
	The input message is not in the predefined topic or domain.
	The closed Q&A model fails to generate a response.

Use Case 2: User Sends Open Domain Q&A Input Message

Table 2: Use Case 2

Use Case 2	User sends open domain Q&A input message
Description	The user sends input message containing specific query within that can be answered across various domains or topics
Primary Actors	User, System
Preconditions	The system is running and has a open domain Q&A model available
Flow of events	1.The user enters a question or query without specifying a domain or a topic
	2.The system receives the input message
	3.The system uses an open Q&A model to process the query.
	4.The open domain Q&A model generates a relevant response based on the query.
Postconditions	The system delivers the response to the user.
Exceptions	The open domain Q&A model is not available
	The open Q&A model fails to generate a response.

3.1.2 Deployment Architecture

The chat bot is a pluggable widget, it easily gets plugged in any website or the system. A user can chat with it to gain any related information. The user can send a query to be processed via two different models that they can easily switch between according to their needs. The software sends requests to the web server depending on the model selected either open domain QA Model or intent classification model. For the intent classification model, the web server then searches for the query in the database and sends back appropriate responses based on the query. For the open domain model, Retriever-Reader pipelines perform what's known as "open-domain extractive question

answering", which will give suitable responses to the user queries. After getting the required answer the result is sent to the server and the server sends it back to the user.

Here, the NLP model plays the vital role that is described below.

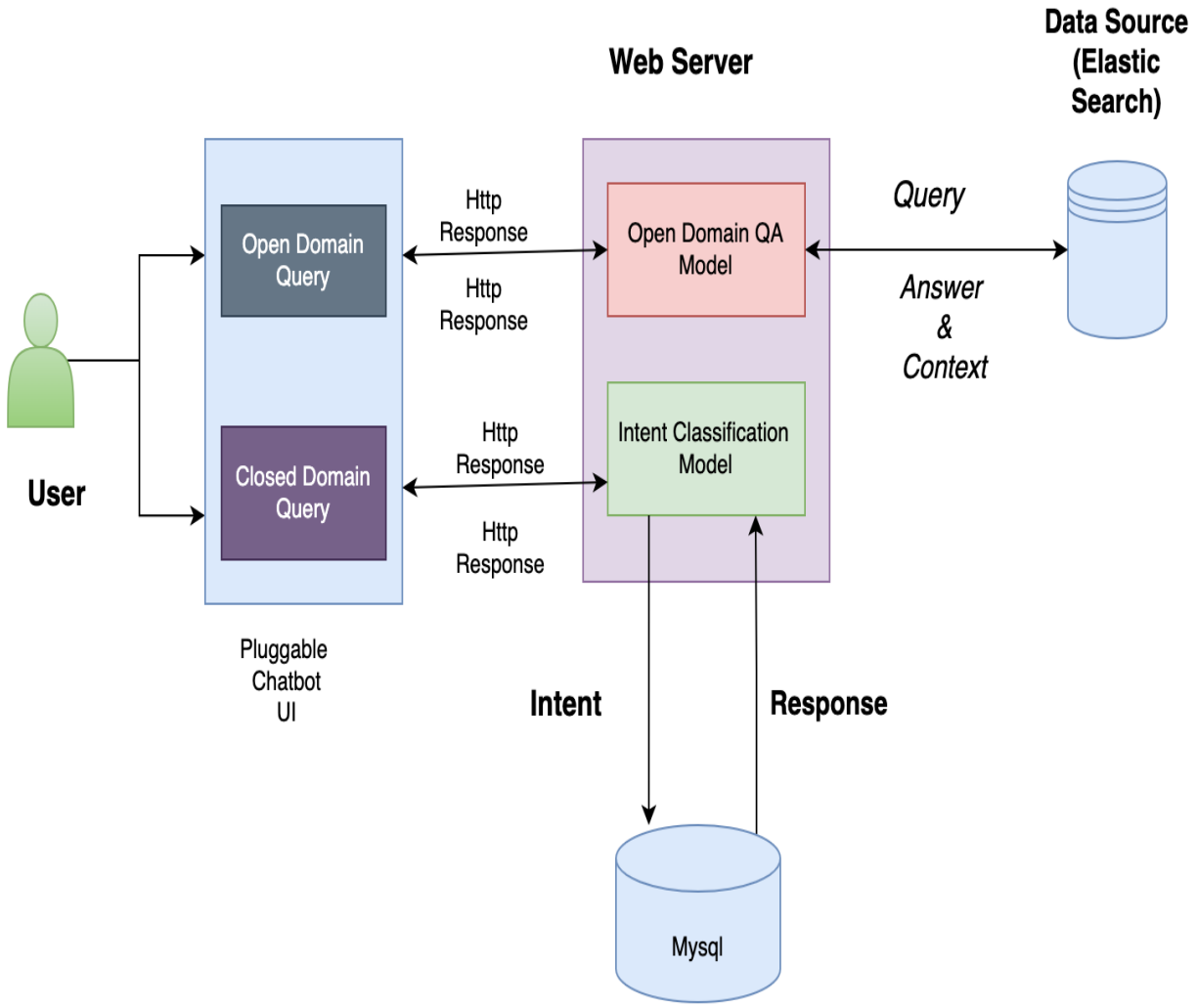

Figure 3: Deployment Architecture

3.1.3 Sequence Diagram

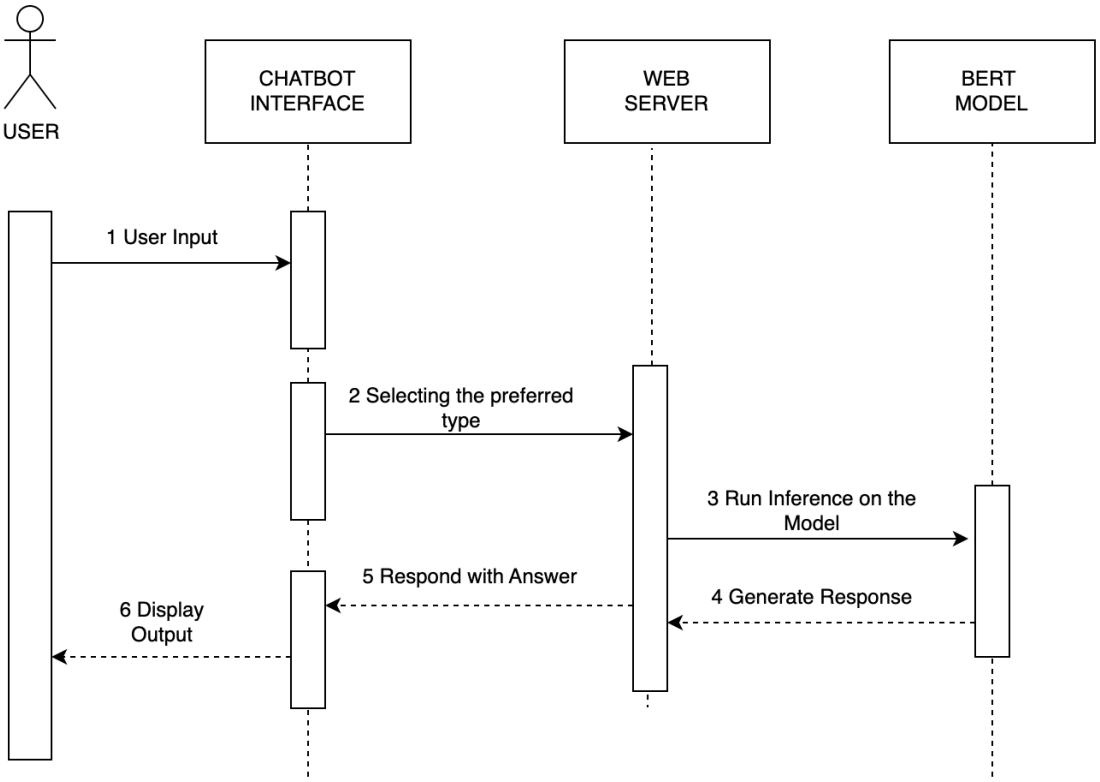

Figure 4: Sequence Diagram

The user enters a query into the chatbot interface by selecting the preferred model type based on the query. The user can easily switch between the two models where one is specifically trained for customer support inquiries and the other is conversational and suitable for providing general information. Once the user submits their query, the web server receives the request and starts processing it to prepare it for searching. For open domain model, once the relevant documents are obtained, the web server generates appropriate responses based on the retrieved information which involves extracting specific fields from the documents, formatting the response, or composing a concise answer. For a [9] closed domain model, after finding relevant information in the database, the web server searches and retrieves appropriate responses based on the user's query. Finally, the web server sends the generated responses back to the user through the chatbot interface. This process enables the chatbot to provide accurate and tailored information to the user.

3.2 Tools and Techniques

Table 3: Tools and Techniques

TOOLS	USAGE
React	Frontend UI
Fast API	Web Server
MySql	Database
Matplotlib	Graph Plot
Tensorflow	Deep Learning Model
Elastic Search	Document Store
Haystack	Pipeline
Docker	Container Virtualization

3.3 Data Source

3.3.1 Intent model Data Source

The data set to be used in this project was first published in the paper “Snips Voice Platform: an embedded Spoken Language Understanding system for private-by-design voice interface” and is freely available in Github.

Table 4: Intent Model Data Source

S.N	Text	Intent
0	Listen to westbam album allergic on google music	PlayMusic
1	Add step to me to the 50 clasicos playlist	AddToPlaylist
2	I give the current textbook a rating value of	RateBook
3	Play the song little robin redbreast	PlayMusic
4	Please add iris dement to my playlist this is	AddToPlaylist
5	What will be the weather	GetWeather
6	Play a popular chant by brian Epstein	PlayMusic
7	Find a fishy story	SearchScreeningEvent
8	Book a spot for 3 in mt	Book Restaurant

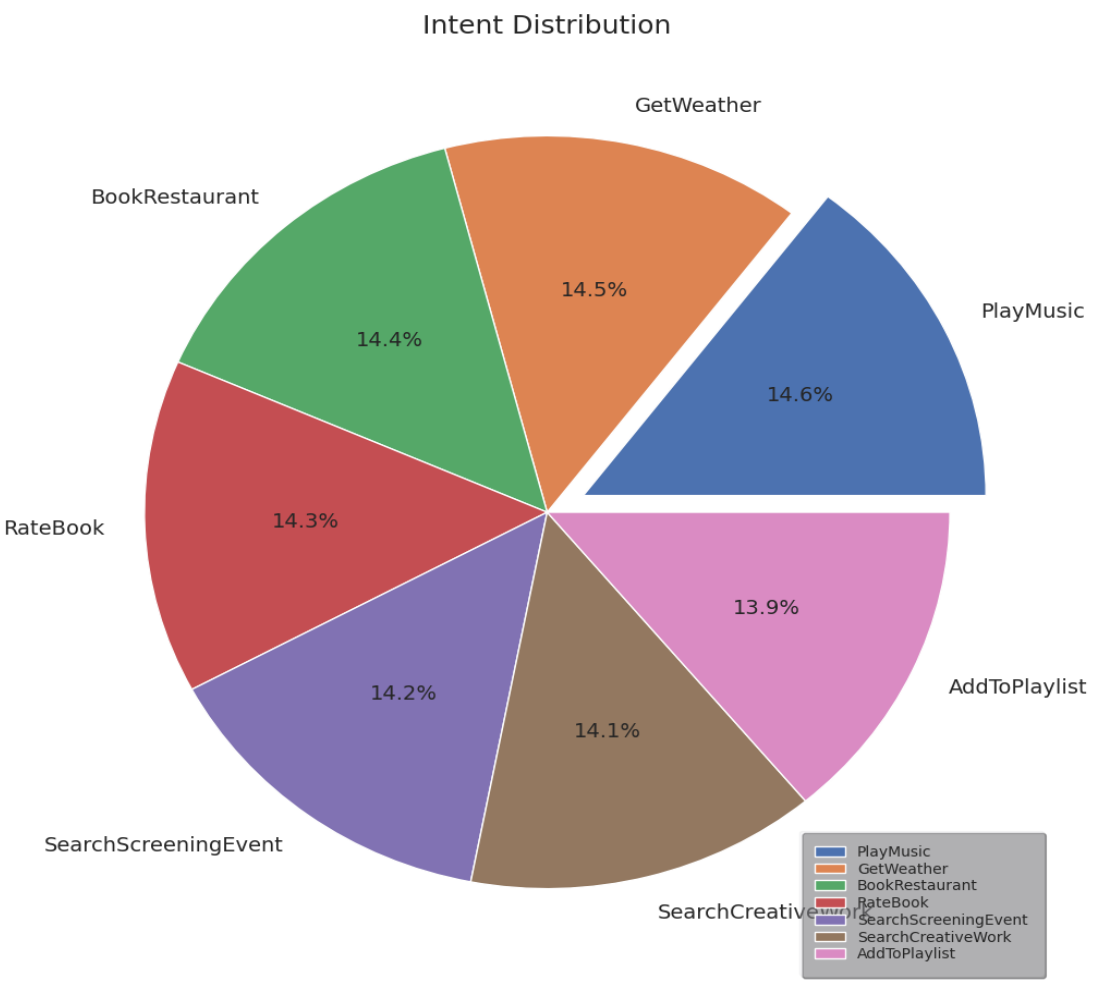

Figure 5: Intent Distribution chart

3.3.2 Q&A SQuAD Model Data Source

The Stanford Question Answering Dataset (SQuAD) is a collection of question-answer pairs derived from Wikipedia articles. In SQuAD, the correct answers to questions can be any sequence of tokens in the given text. Because the questions and answers are produced by humans through crowdsourcing, it is more diverse than some other question-answering datasets. SQuAD 1.1 contains 107,785 question-answer pairs on 536 articles. SQuAD2.0 (open-domain SQuAD, SQuAD-Open), the latest version, combines the 100,000 questions in SQuAD1.1 with over 50,000 un-answerable questions written adversarially by crowd workers in forms that are similar to the answerable ones.

Table 5: SQuAD Model Data Source

Field	**Description**
Id	tensor containing article id.
Title	tensor containing title
Context	tensor containing context from the title
Question	tensor containing the question
Text	tensor containing the text
answer_start	tensor that contains the starting index of the answer
is_impossible	A label that represents whether answering is possible or impossible. If it is impossible, the label represents the value 1 for True. If not impossible label represents the value o for False.

Sample SQuad Format

```
{
  "data": [
    {
      "paragraphs": [
        {
          "context": "The quick brown fox jumps over the lazy dog.",
          "qas": [
            {
              "question": "What does the fox jump over?",
              "id": "q1",
              "answers": [
                {
                  "text": "the lazy dog",
                  "answer_start": 32
                }
              ]
            }
          ]
        }
      ],
      "title": "Example"
    }
  ],
  "version": "2.0"
```

3.3.3Data Split

Table 6: Data Split

Training Set	86821 samples
Testing Set	20302 samples

3.4 Algorithm

3.4.1 Bert Model

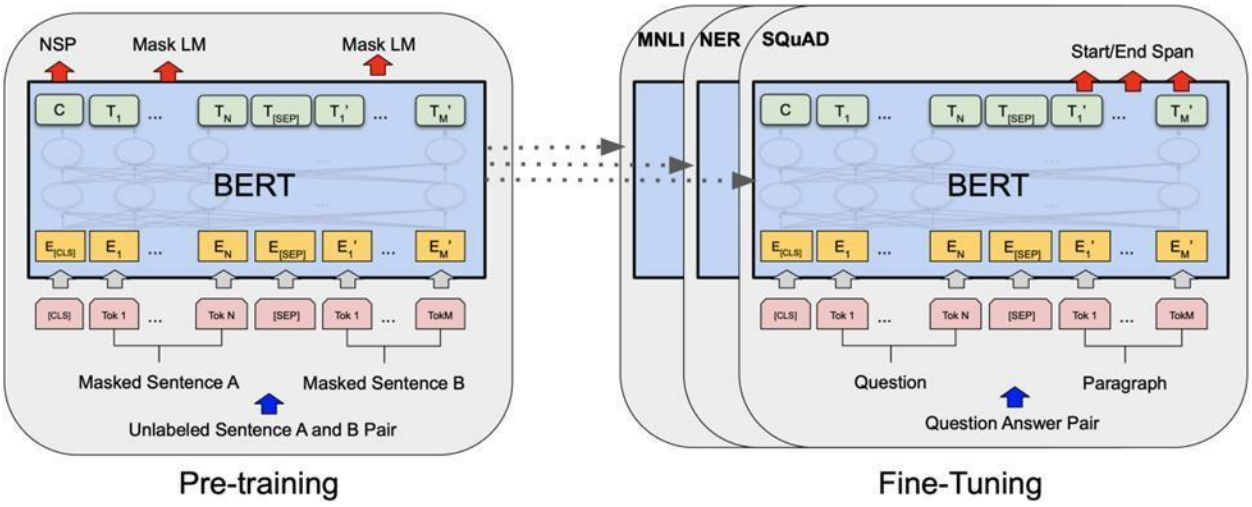

Figure 6: BERT Model

BERT is a method of pre-training language representations. [4] BERT is the first deeply bidirectional, unsupervised language representation, pre-trained using only a plain text corpus.

It involves two steps:

- **Pre training**

  BERT models are pre trained using the enormous amount of unannotated text on the web (Wikipedia).

- **Fine tuning**

  The training results can be applied to other Natural Language Processing (NLP) tasks, such as question answering and sentiment analysis. the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications.

3.4.2 Intent Classification Flow Model

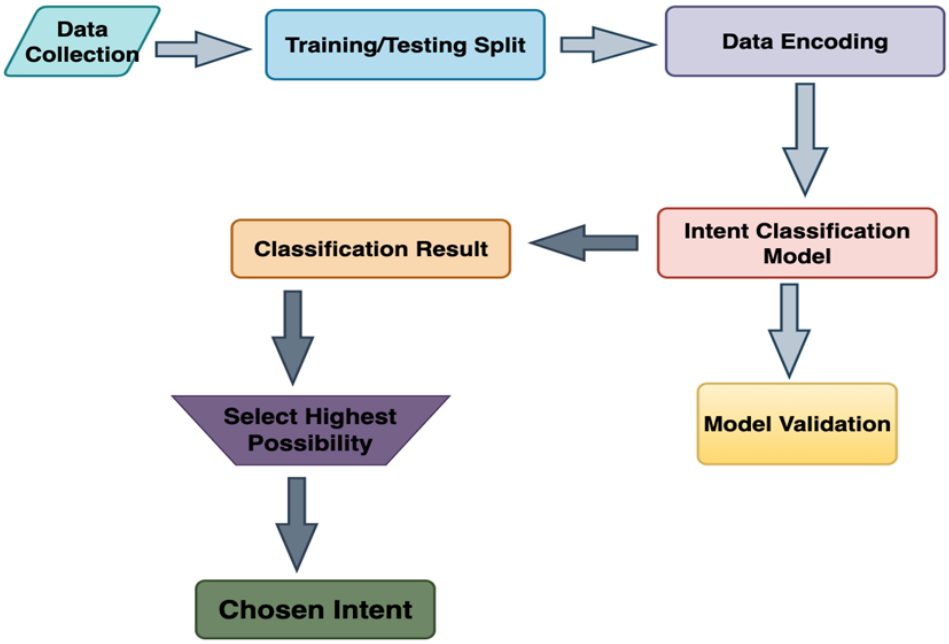

Figure 7: Intent Classification Model

The intent flow classification stage is a crucial component of intent recognition systems in natural language processing applications. It involves several stages, starting from data collection to selecting the most probable intent. Here is a diagram explaining the various stages:

- **Data Collection:**

  The first stage of the intent flow classification is data collection. This involves gathering a diverse and representative dataset that encompasses different user intents. The dataset should include labeled examples of user queries or utterances along with their corresponding intents.

- **Train/Test Split:**

  In the second stage, the collected dataset is divided into training and testing subsets. [7]The training subset is used to train the intent classification model, while the testing subset is employed to evaluate the model's performance and generalization capabilities.

- **Data Encoding:**

  The third stage involves data encoding, where the text data is transformed into numerical representations that can be understood by machine learning models. This process typically involves techniques such as word embedding or one-hot encoding to convert textual data into numerical vectors.

- **Intent Classification Model**:

  The fourth stage is the core of the intent flow classification, where an intent classification model is trained using the encoded data. This stage can employ various machine learning algorithms, such as logistic regression, support vector machines, or neural networks, to learn the mapping between the input queries and their corresponding intents.

- **Classification Result and Model Validation:**

  In the fifth stage, the trained intent classification model is used to predict the intents of unseen queries from the testing subset. The classification results are compared against the ground truth labels to evaluate the model's accuracy, precision, recall, and other performance metrics. Model validation techniques, such as cross-validation or holdout validation, can be applied to ensure reliable results.

- **Select Highest Possibility:**

  After the intent classification model predicts the intents for the testing queries, this sixth stage involves selecting the intent with the highest probability or confidence score as the predicted intent. This selection is based on the model's output probabilities for each intent class.

- **Chosen Intent:**

  The final stage of the intent flow classification is determining the chosen intent based on the intent with the highest probability. The chosen intent is the final output of the intent recognition system and represents the predicted intent for a given user query or utterance.

  The intent flow classification stage follows a systematic process, starting from data collection and train/test split, to data encoding, training the intent classification model, evaluating the model's performance, selecting the highest probability intent, and determining the chosen intent. This

diagram provides an overview of the stages involved in accurately recognizing user intents, enabling effective natural language understanding and interaction in various applications, such as chatbots, virtual assistants, and customer support systems.

3.4.3 Bert Configuration

- hidden_size: Dimensionality of the encoder layers and the pooler layer.
- hidden_act — The non-linear activation function (function or string) in the encoder and pooler. If string, "gelu", "relu", "silu" and "gelu_new" are supported.
- initializer_range — The standard deviation of the truncated_normal_initializer for initializing all weight matrices.
- vocab_size — Vocabulary size of the BERT model. Defines the number of different tokens that can be represented by the inputs_ids passed when calling BertModel or TFBertModel.
- hidden_dropout_prob— The dropout probability for all fully connected layers in the embeddings, encoder, and pooler.
- num_attention_heads — Number of attention heads for each attention layer in the Transformer encoder.
- type_vocab_size— The vocabulary size of the token_type_ids passed when calling BertModel or TFBertModel.
- max_position_embeddings — The maximum sequence length that this model might ever be used with. Typically set this to something large just in case (e.g., 512 or 1024 or 2048).
- num_hidden_layers — Number of hidden layers in the Transformer encoder.
- intermediate_size — Dimensionality of the “intermediate” (often named feed-forward) layer in the Transformer encoder.
- attention_probs_dropout_prob (float, optional, defaults to 0.1) — The dropout ratio for the attention probabilities

Table 7: Bert Configuration

Hidden_size	768
Hidden_act	Gelu
Initializer_range	0.02
Vocab_size	30522
Hidden_dropout_prob	0.1
Num_attention_heads	12
Type_vocab_size	2
Max_position_embeddings	512
Num_hidden_layers	12
Intermediate_size	3072
Attention_probs_dropout_prob	0.1

3.4.4. BERT Training strategies

- **Masked Language Modeling (MLM)**

  Masking out some of the words in the input and then condition each word bidirectionally to predict the masked words. In the pre-training phase, the model is fed with a large corpus of text and learns to predict the masked words in the input sequence.

  For example, if the sentence is “The cat in the [MASK] is black”, the model needs to predict the missing word based on the context of the sentence. In this case, the answer could be “hat”, “box” etc. based on the model’s training data.

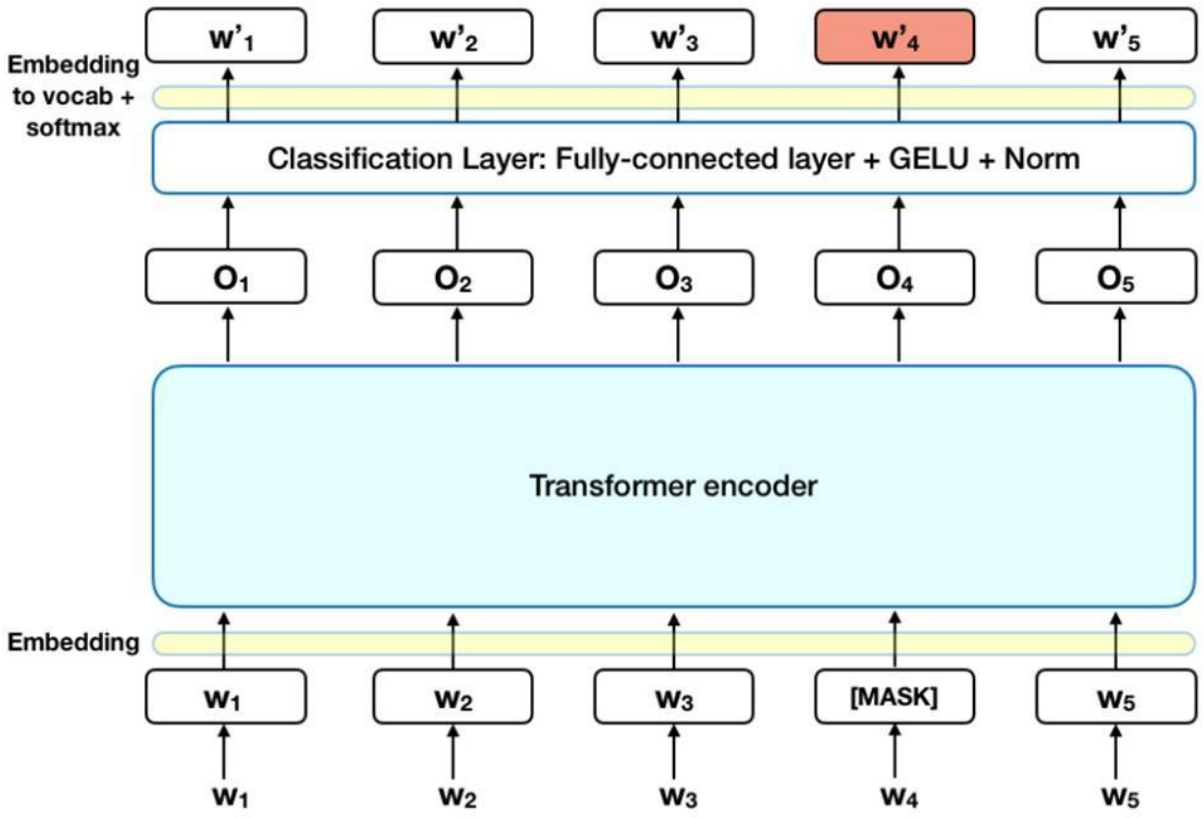

Figure 8: MLM Workflow

- **Next Sentence Prediction (NSP)**

  This technique involves training a model to predict whether a given sentence follows another sentence or not. In the pre-training phase, the model is presented with pairs of sentences, and the goal is to predict whether the second sentence is a continuation of the first one or not. The model is trained to learn the semantic relationship between the two sentences, such as the cause and effect, implication, or contradiction between them. For example:

Sentence A = `The man went to the store.`
Sentence B = `He bought a gallon of milk.`
Label = `IsNextSentence`

Sentence A = `The man went to the store.`
Sentence B = `Penguins are flightless.`
Label = `NotNextSentence`

3.4.5. Data Preprocessing

- **Tokenize the sentence**

  Tokenization involves breaking down a sentence into smaller units called tokens. For example, the sentence “I love cats” might be tokenized into [“I”,”love”,”cats”]

- **Add special tokens [CLS] at beginning and [SEP] at end**

  BERT requires special tokens to mark the beginning and end of each input sequence. The [CLS] token is added at the beginning of the sentence, and the [SEP] token is added at the end.

- **Make sentence of same length by padding**

  BERT models typically require fixed-length input sequences. However, input sentences can have different lengths. To handle this, padding is applied to make all sentences the same length by adding a special padding token, usually [PAD], to the end of shorter sentences. For example, if the maximum sequence length is set to 10 and the sentence "I love cats" has 3 tokens, it will be padded as ["I", "love", "cats", [PAD], [PAD], [PAD], [PAD], [PAD], [PAD], [PAD]].

- **Create attention mask**

  Attention masks are used to indicate which tokens in the input sequence the model should pay attention to and which ones should be ignored. The attention mask has the same length as the input sequence and contains 1s for tokens that should be attended to and 0s for padded tokens that should be ignored. In the above example, the attention mask would be [1,1,1,0,0,0,0,0,0,0]

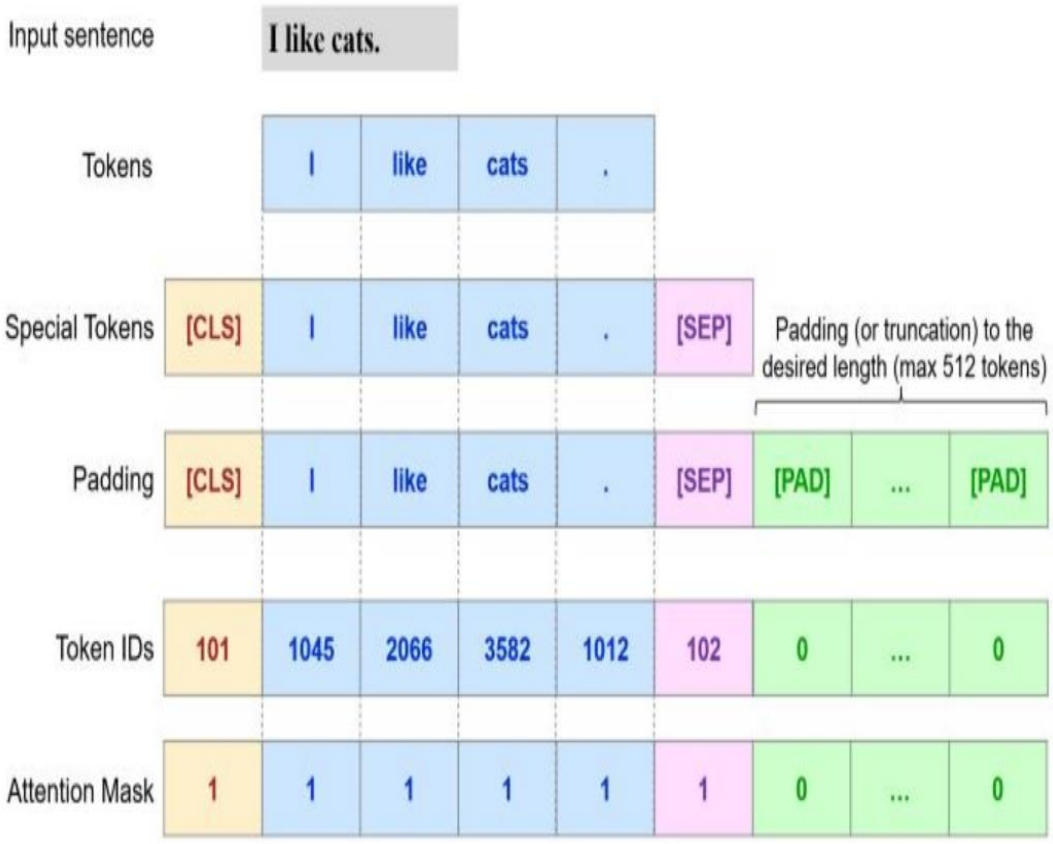

Figure 9: Data Pre-Processing

3.4.6. Open Q&A Architecture

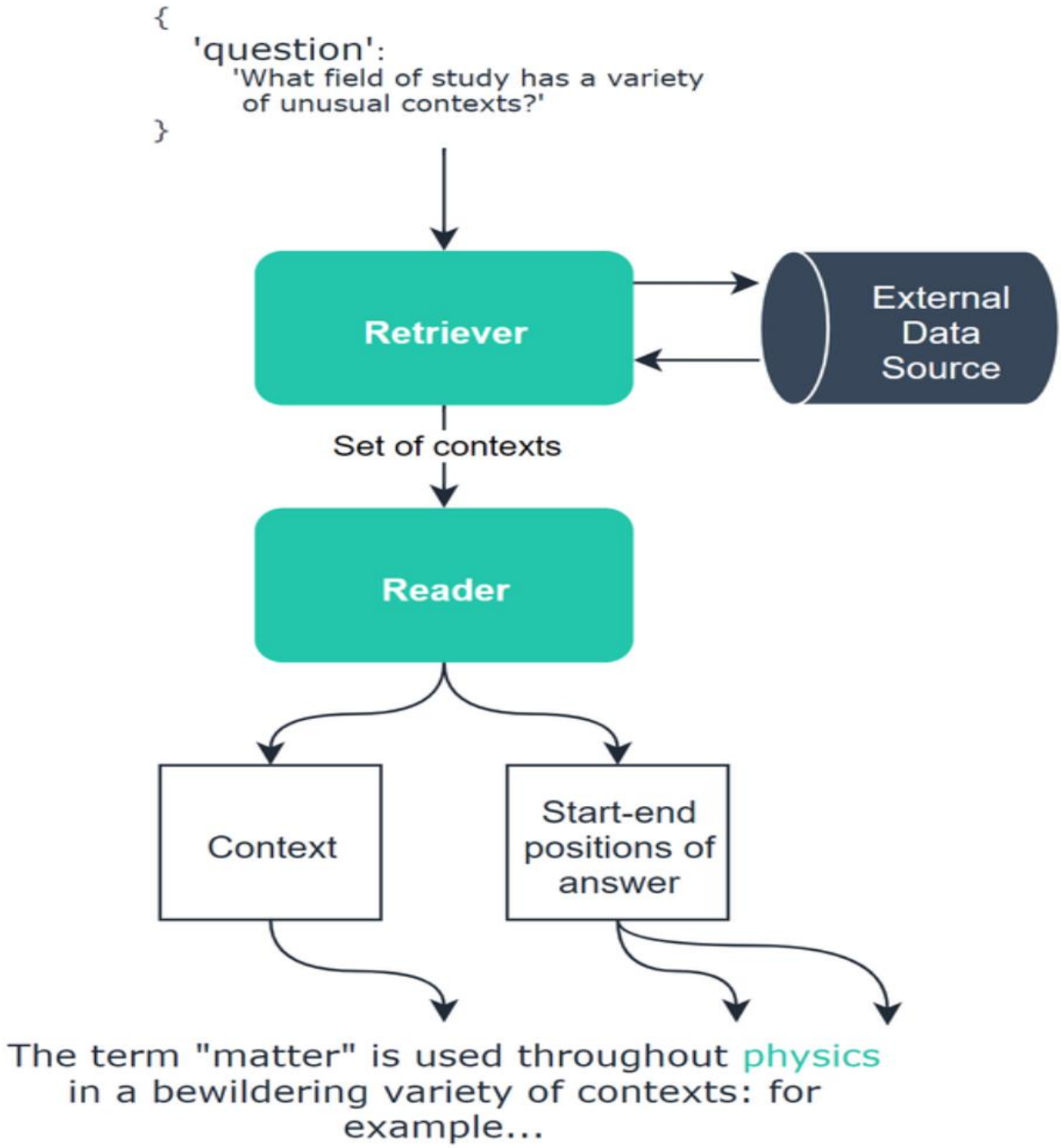

Figure 10: Open Q&A Architecture

3.4.7. Open Q&A Model

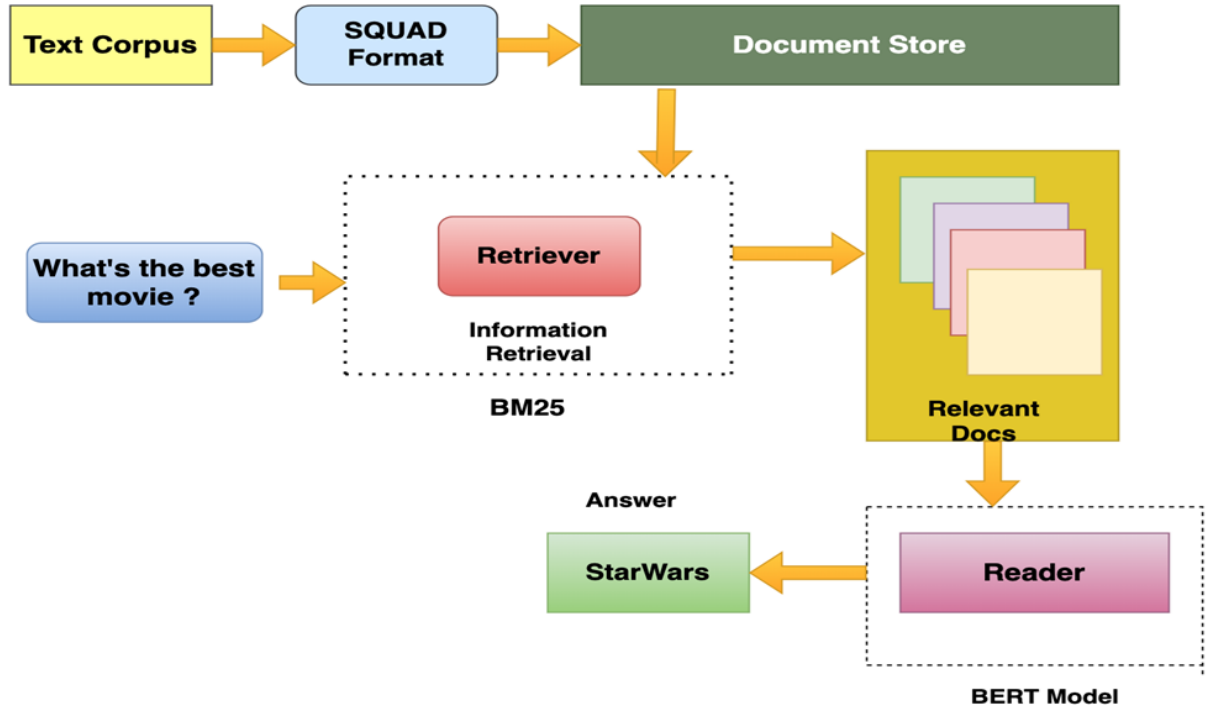

Figure 11: BM25

The Extractive Question-Answering (QA) pipeline model is a framework that aims to answer questions based on a given text corpus. It involves several key components, including the text corpus, the SQuAD format, a BM25 retriever, relevant document selection, and a BERT-based model. This article explores the architecture and functionalities of each component in the pipeline model.

Text Corpus:

The text corpus is a collection of documents or passages from which the QA pipeline extracts answers. It can be a diverse range of sources, including books, articles, or web pages. The text corpus serves as the foundation for the QA system, providing the necessary information to generate answers to questions.

SQuAD Format:

The SQuAD (Stanford Question Answering Dataset) format is a widely adopted standard for QA tasks. It represents the text corpus in a structured manner, with each document consisting of paragraphs, and each paragraph containing a list of questions and their corresponding answers. This format facilitates the training and evaluation of QA models by providing ground truth answers for the questions.

BM25:

BM25 is a term-based ranking model that aims to provide accurate and relevant search results by scoring documents based on their term frequencies and document lengths. It can be mathematically expressed as

$$\text{score}(D,Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)},$$

Term Frequency (TF): TF refers to the number of times a particular term appears in a document. However, BM25 uses a modified term frequency that takes into account saturation effects to prevent overemphasizing heavily repeated terms.

Inverse Document Frequency (IDF): IDF measures the importance of a term in the entire corpus. It assigns higher weights to terms that are rare in the corpus and lower weights to terms that are common. IDF is calculated using the formula: IDF = log((N — n + 0.5) / (n + 0.5)), where N is the total number of documents and n is the number of documents containing the term.

Document Length Normalization: BM25 incorporates document length normalization to address the impact of document length on relevance scoring. Longer documents tend to have more occurrences of a term, leading to potential bias. Document length normalization counteracts this bias by dividing the term frequency by the document's length and applying a normalization factor.

Query Term Saturation: BM25 also includes a term saturation function to mitigate the impact of excessively high term frequencies. This function reduces the effect of extremely high term

frequencies on relevance scoring, as very high frequencies often correspond to less informative terms.

Relevant Document Selection:

Once the BM25 retriever generates a ranked list of documents, the relevant document selection component identifies the most promising passages that may contain the answer to the question. This step helps narrow down the search space and reduces the computation required for subsequent processing. The selection process typically involves selecting the top-k passages based on their BM25 scores.

BERT-based Model:

The BERT (Bidirectional Encoder Representations from Transformers)-based model is employed for question answering within the extractive QA pipeline. BERT models are pre-trained on large amounts of data and possess a deep understanding of language semantics. They can effectively encode input text and generate meaningful representations for downstream tasks like question answering.

The BERT model takes as input the selected relevant passages and the user's question. It encodes the text into contextualized representations using transformer-based architectures. The model then employs attention mechanisms to identify the most salient information relevant to the question within the selected passages.

Output:

The output of the extractive QA pipeline is the answer to the user's question. The BERT model processes the relevant passages and the question to identify the most suitable answer span within the passages. The model selects the answer span that maximizes a score based on the contextual information provided by the BERT-based architecture.

The answer span is typically a contiguous sequence of words within the selected passages. It is extracted directly from the text rather than being generated. This extractive approach ensures that the answer is grounded in the original text and reduces the risk of generating incorrect or nonsensical answers.

The extractive QA pipeline model comprises several interconnected components that work together to generate answers to user questions based on a given text corpus. It leverages the SQuAD format to structure the corpus, utilizes the BM25 retriever to rank and retrieve relevant documents, employs relevant document selection to narrow down the search space, and utilizes BERT-based models to extract the answer span from the relevant passages. This pipeline architecture enables accurate and contextually relevant question answering, making it a valuable tool in various domains such as information retrieval, customer support, and educational applications.

4. Performance, Validation and Analysis:

4.1 Confusion Matrix:

It is a table that summarizes the performance of a classification model by displaying the counts of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) predictions.

- **PRECISION**: It measures the proportion of predicted positive instances out of the total instances predicted as positive. It is calculated as the ratio of true positives (TP) to the sum of true positives and false positives (FP).
  Precision=TP/(TP+FP)

- **RECALL**: It measures the proportion of correctly predicted positive instances out of the total actual positive instances. It is calculated as the ratio of true positives (TP) to the sum of true positives and false negatives (FN). Recall=TP/(TP+FN) F1

- **SCORE:** The F1 score ranges from 0 to 1, with 1 being the best performance. It is calculated as F1 Score=2*(precision*Recall)/ (Precision + Recall). LOSS CURVE: A loss curve represents the loss of the model during training over epochs or iterations. It shows how the loss decreases as the model learns and updates its parameters. ACCURACY

- **CURVE:** It is a graphical representation of the model's accuracy over different threshold or decision boundaries. It shows how the model's accuracy varies as the threshold for classification is changed. By adjusting the threshold, you can trade of between precision and recall, as well as customize the models behavior based on the specific requirements of the problem.

- **SUPPORT:** The support metric represents the total number of instances belonging to a specific class, regardless of whether they were correctly or incorrectly classified. It is typically displayed as a row or column in the confusion matrix, showing the total count of instances for each class.

Table 8: Performance Table

Intent	**Precision**	**Recall**	**F1-score**	**Support**
PlayMusic	0.88	0.92	0.90	86
AddToPlaylist	0.98	0.98	0.98	124
RateBook	1.00	1.00	1.00	80
SearchScreeningEvent	0.95	0.91	0.93	107
Book Restaurant	0.99	1.00	0.99	92
GetWeather	1.00	0.99	1.00	104
SearchCreativeWork	0.87	0.88	0.87	107

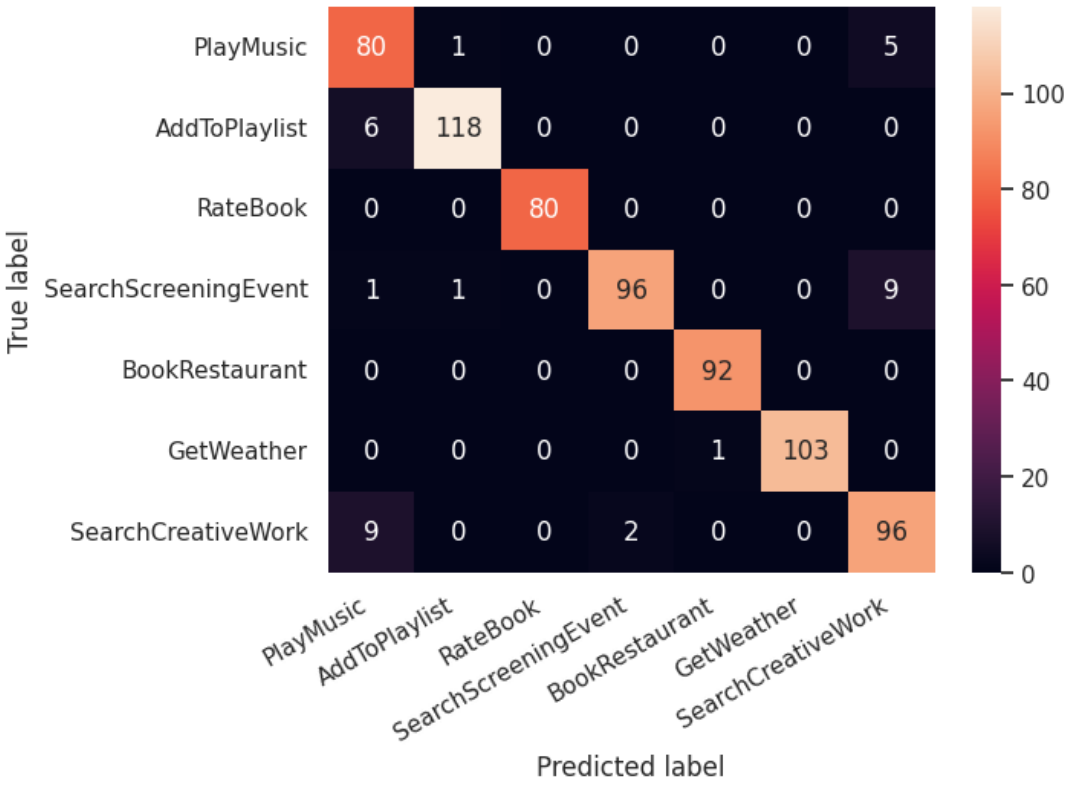

Figure 12: Confusion Matrix

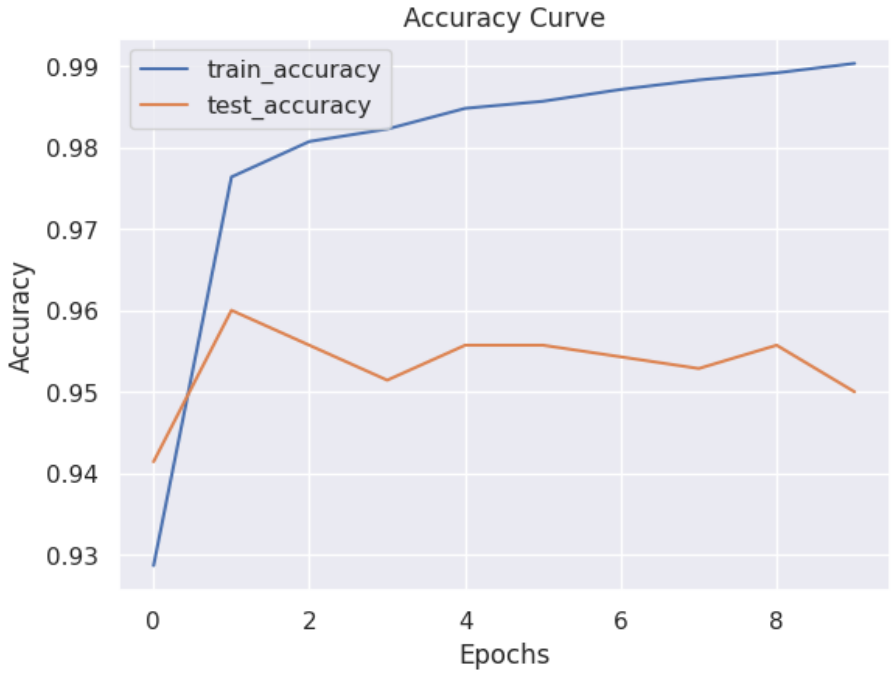

Figure 13: Accuracy Curve

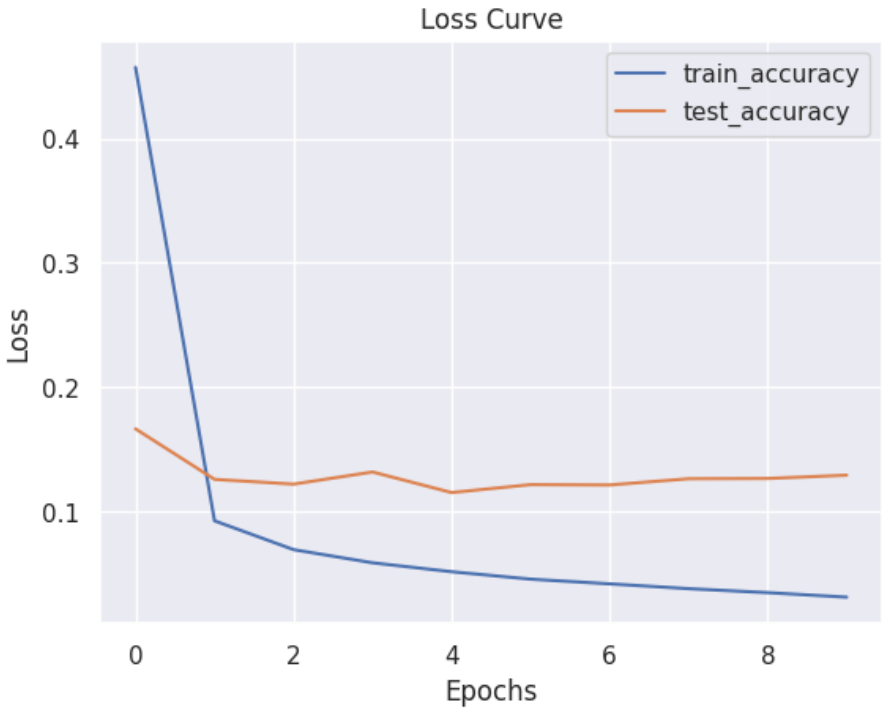

Figure 14: Loss Curve

4.2 Performance of Q&A model

ROGUE Metric for Q&A Model

ROUGE stands for Recall-Oriented Understudy for Gisting Evaluation. ROUGE is actually a set of metrics, rather than just one.

ROUGE-N

ROUGE-N measures the number of matchings 'n-grams' between our model-predicted answer and a 'reference'.

An n-gram is simply a grouping of tokens/words. A unigram (1-gram) would consist of a single. A bigram (2-gram) consists of two consecutive words:

Original: "the quick brown fox jumps over"

Unigrams: ['the', 'quick', 'brown', 'fox', 'jumps', 'over']

Bigrams: ['the quick', 'quick brown', 'brown fox', 'fox jumps', 'jumps over']

Trigrams: ['the quick brown', 'quick brown fox', 'brown fox jumps', 'fox jumps over']

The reference in our case is our true answer.

With ROUGE-N, the N represents the n-gram that we are using. For ROUGE-1 we would be measuring the match-rate of unigrams between our model output and reference.

ROUGE-2 and ROUGE-3 would use bigrams and trigrams respectively.

Table 9: Rogue Table

ROGUE	**F1**	**Precision**	**Recall**
ROGUE-1	0.759	0.733	0.8
ROGUE-2	0.549	0.521	0.6

5. Project Output

- The project output is a pluggable chatbot widget.
- Switchable chatbot with open and closed Q&A model.
- A flexible and scalable chatbot system that provides customization options for suitable types of servers.
- System that provides 24/7 customer support.

6. Task and Time Schedule

6.1. V model stages

The V model is favored over other software development models because of its emphasis on clear and structured stages that align development activities with corresponding testing activities. The V model typically consists of the following stages:

- Requirements Gathering: The project's requirements are collected and documented in detail, ensuring a comprehensive understanding of what the software needs to achieve.
- System Design: The high-level system architecture and design are defined, mapping the requirements to system components and establishing the overall system structure.
- Subsystem Design: The system design is further refined into detailed subsystem designs, specifying the design of individual components or modules.
- Unit Implementation: The actual coding or development of the software units (modules) takes place at this stage. Each unit is implemented based on the corresponding subsystem design.
- Unit Testing: The implemented units are tested in isolation to ensure that they function as intended and meet the specified requirements.
- Integration: The individual units are integrated to form the complete system, verifying that they work together correctly and comply with the system design.
- System Testing: The integrated system is thoroughly tested to ensure that it meets all functional and non-functional requirements, identifying and resolving any defects or discrepancies.
- User Acceptance Testing (UAT): The system is tested by end-users or stakeholders to validate its usability, functionality, and compliance with their needs and expectations.
- Deployment: The fully tested and approved system is deployed in the production environment or delivered to the end-users.
- Maintenance: Once the system is deployed, it enters the maintenance phase, where updates, bug fixes, and enhancements are made as needed to ensure its ongoing functionality and usability.

6.2. Task Division

Table 10: Task Division

Name	Task Division
Sadhika Kasaju	Documentation, Data Collection, Backend, Testing
Dishan Shrestha	Documentation, Frontend, Database
Simran Tamrakar	Documentation, Backend, ML engineer
Rinku Deuja	Documentation, Data Collection, Testing, Frontend

6.3 Time Schedule and Gantt Chart

Table 11: Time Schedule

Task	Start Date	End Date	Duration
Dataset Collection	5/10/2023	5/17/2023	7
Dataset Pre-Processing	5/15/2023	5/30/2023	15
Model Development	5/31/2023	6/27/2023	27
Testing and Validation	6/20/2023	7/2/2023	12
Documentation	5/15/2023	7/10/2023	56

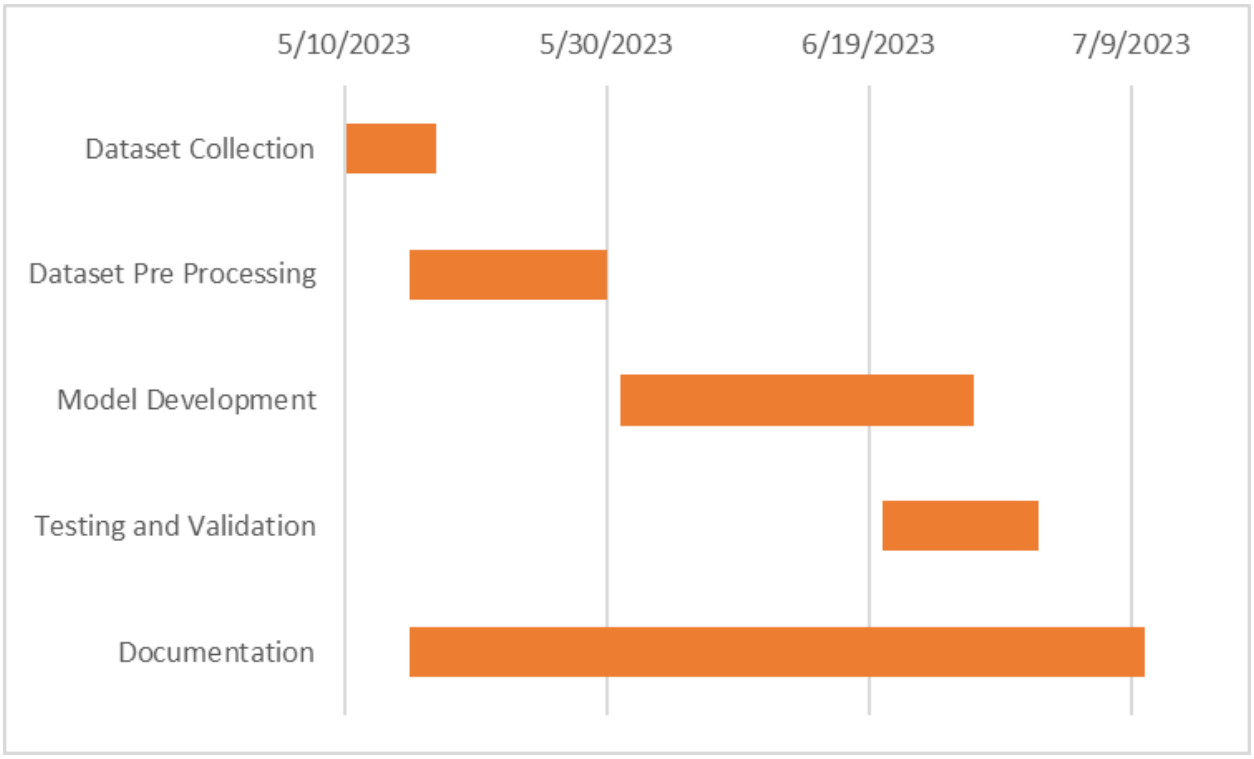

Figure 15: Gantt Chart

7. Conclusion

This project has successfully fine-tuned the BERT model and implemented the model to create a multipurpose chatbot application that can be easily plugged into any pre-existing web application. We have managed to leverage the state of art NLP model to build a chatbot that can work on both labeled and unlabeled data and accurately extract the information based on users' queries.

The pipeline created in the project can be easily customized to any business sector and provide tailored chat-based solutions on a wide variety of textual data. The chatbot that has been designed is highly scalable and has a far superior response rate.

The project also illustrates how a deep learning model can be integrated into the business layer of any web application. Further, the transformer NLP model has been effectively used and deployed to suit different business requirements.

Traditional usage of customer support using text-based communication can be enhanced by adopting the chatbot developed in this project. It will eradicate the problem of delayed response and can make the customer support experience smooth and easy.

The project also uses SQL database and ElasticSearch showcasing how different models of data stores can be used seamlessly to provide custom solutions.

8. Recommendations and Further Works

The project can be further enhanced with the following features:

- The project can be upgraded to a multilingual chatbot system.
- The chatbot can be enhanced to integrate images, gifs, and other interactive multimedia technologies to create better user experience.
- The model can be further fine-tuned to improve accuracy.

9. References

[1] Caldarini, G.; Jaf, S.; McGarry, K. “A Literature Survey of Recent Advances in Chatbots”,15 Jan 2022

[2] Ohose, Ejiro “Building machine learning chatbots”, 21 April 2023,

[3] Mondal, Arnav “Complete guide to build your AI chatbot with NLP in Python”, 25 Oct 2021,

[4] Horev, Rani “BERT: State of the Art NLP Model,Explained”, 11 Aug,2018, BERT: State of the Art NLP Model, Explained - KDnuggets

[5] Kanodia N., Ahmed K., Miao Y. “Question Answering Model Based Conversational chatbot using BERT model and google dialogflow”, 10 Nov 2021,Question Answering Model Based Conversational Chatbot using BERT Model and Google Dialogflow (computer.org)’

[6] Muhammad Bilal, Almazroi A. “Effectiveness of Fine-tuned BERT Model in Classification of helpful and unhelpful online customer review” 29 April 2022, Effectiveness of Fine-tuned BERT Model in Classification of Helpful and Unhelpful Online Customer Reviews | SpringerLink

[7] Bland, Greg. “Train/Test Split and Cross Validation - A Python Tutorial”, 13 Oct 2020,Train/Test Split and Cross Validation - A Python Tutorial - AlgoTrading101 Blog

[8] Brush, Kate, and Jesse Scardina. “What Is a Chatbot and Why Is It Important?”,18 Nov. 2021

[9] Mikio Nakano, Kazunori Komatani “A Framework for Building Closed-Domain Chat Dialogue Systems”,21 July 2020

[10] Ortiz,Sabrina “What is chatGPT and why does it matter?”,26 june 2023, ChatGPT and why does it matter?

[11] Sharma,Upanishad “Snapchat ‘My AI’ Chatbot: What Is It, How Does It Work, How to Use, and More”,26 April 2023, how snapchat my AI works

Appendix

1. Homepage

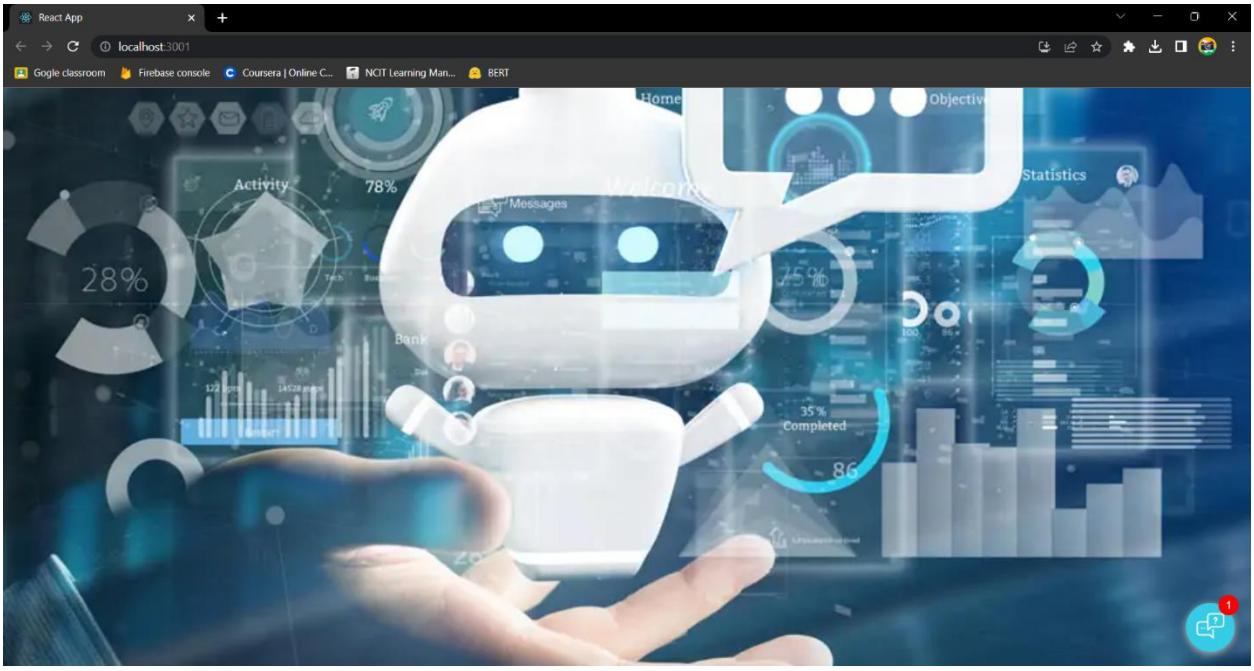

Figure 16: Homepage

2. Chatbot popup

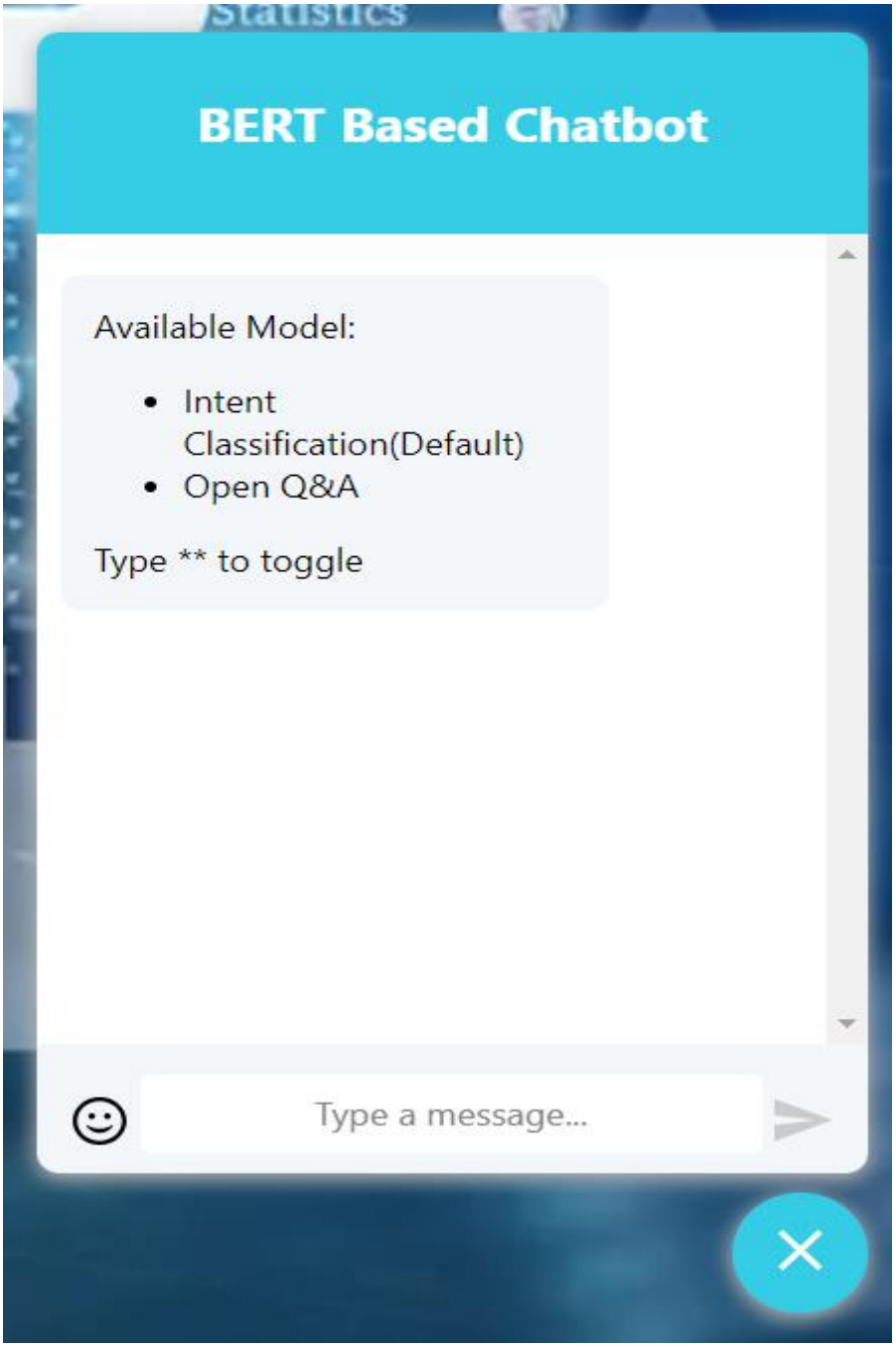

Figure 17: Chatbot popup

3. Intent chats

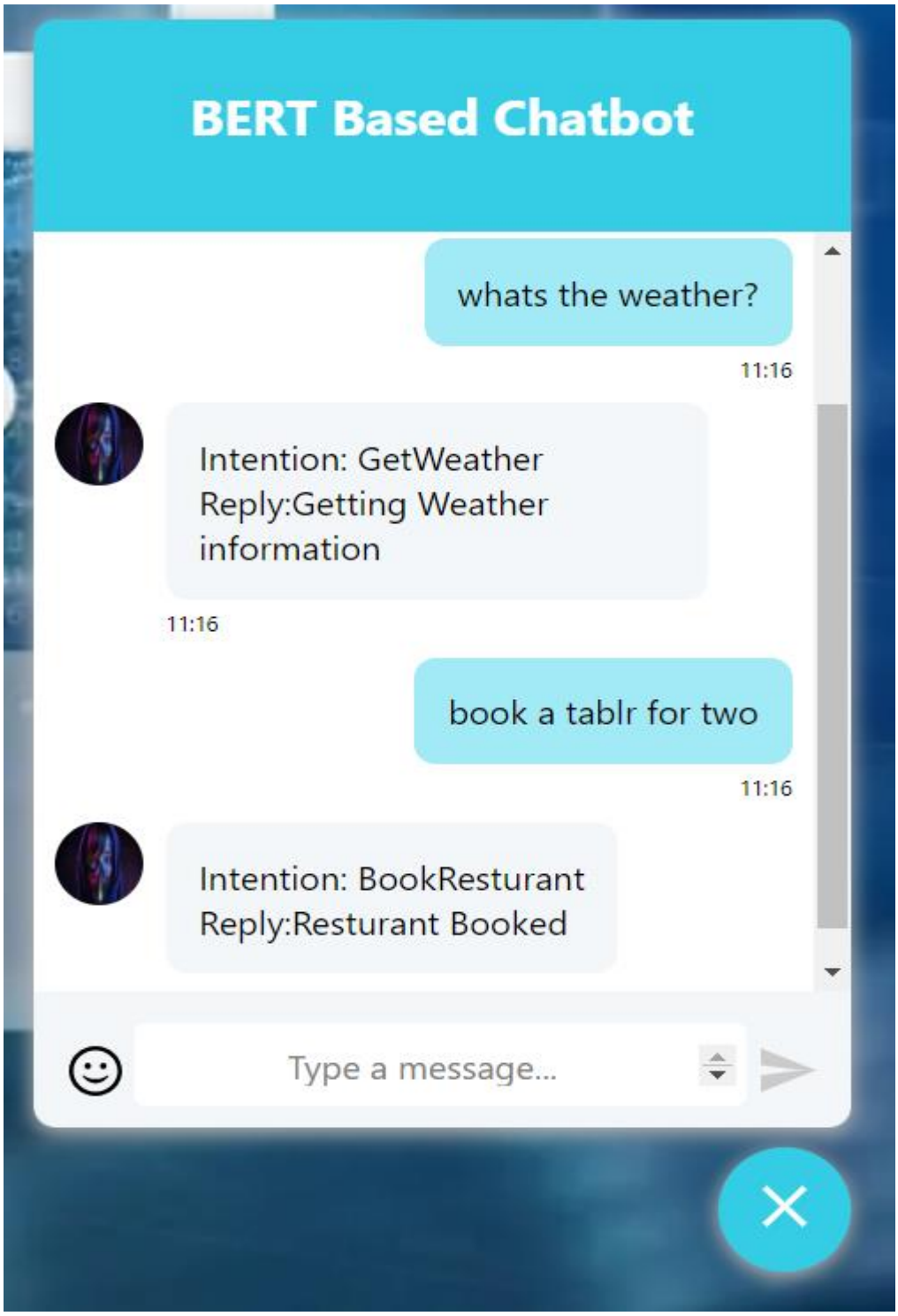

Figure 18: Intent chats

4. Model switched to conversational chat

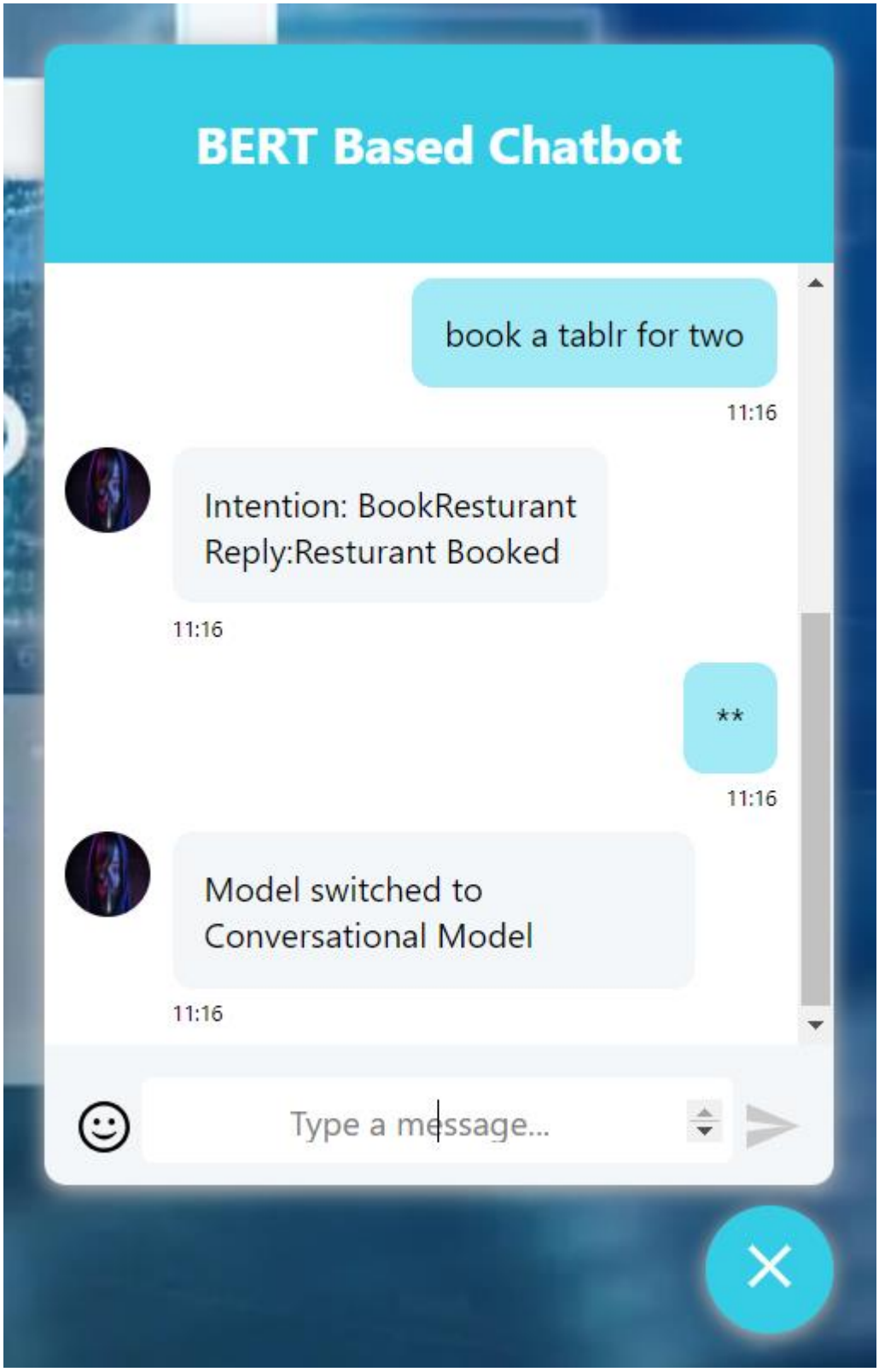

Figure 19: Model switch to Conversational

5. Asking Queries about anything

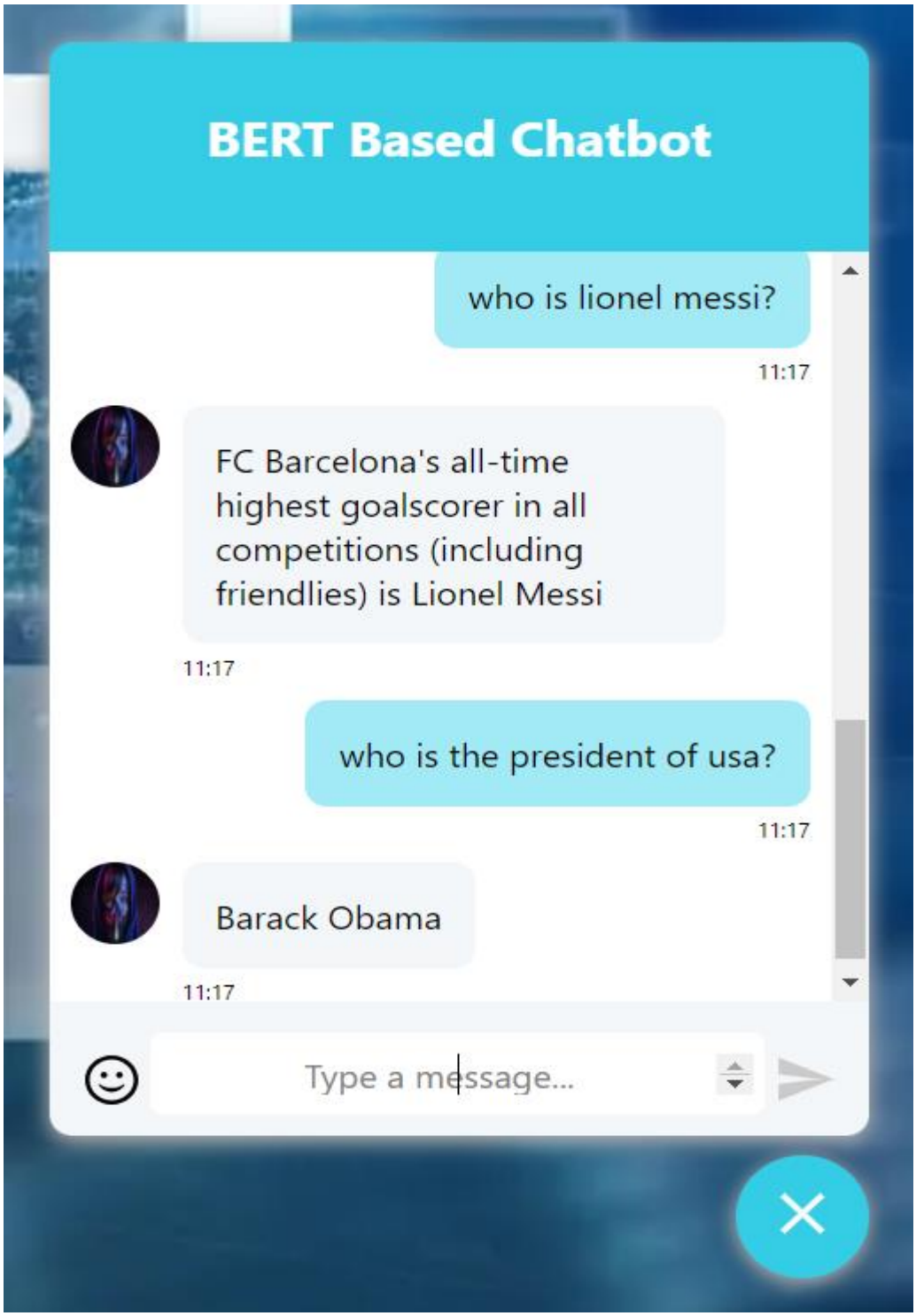

Figure 20: Asking Questions

6. Model switched back to Intent

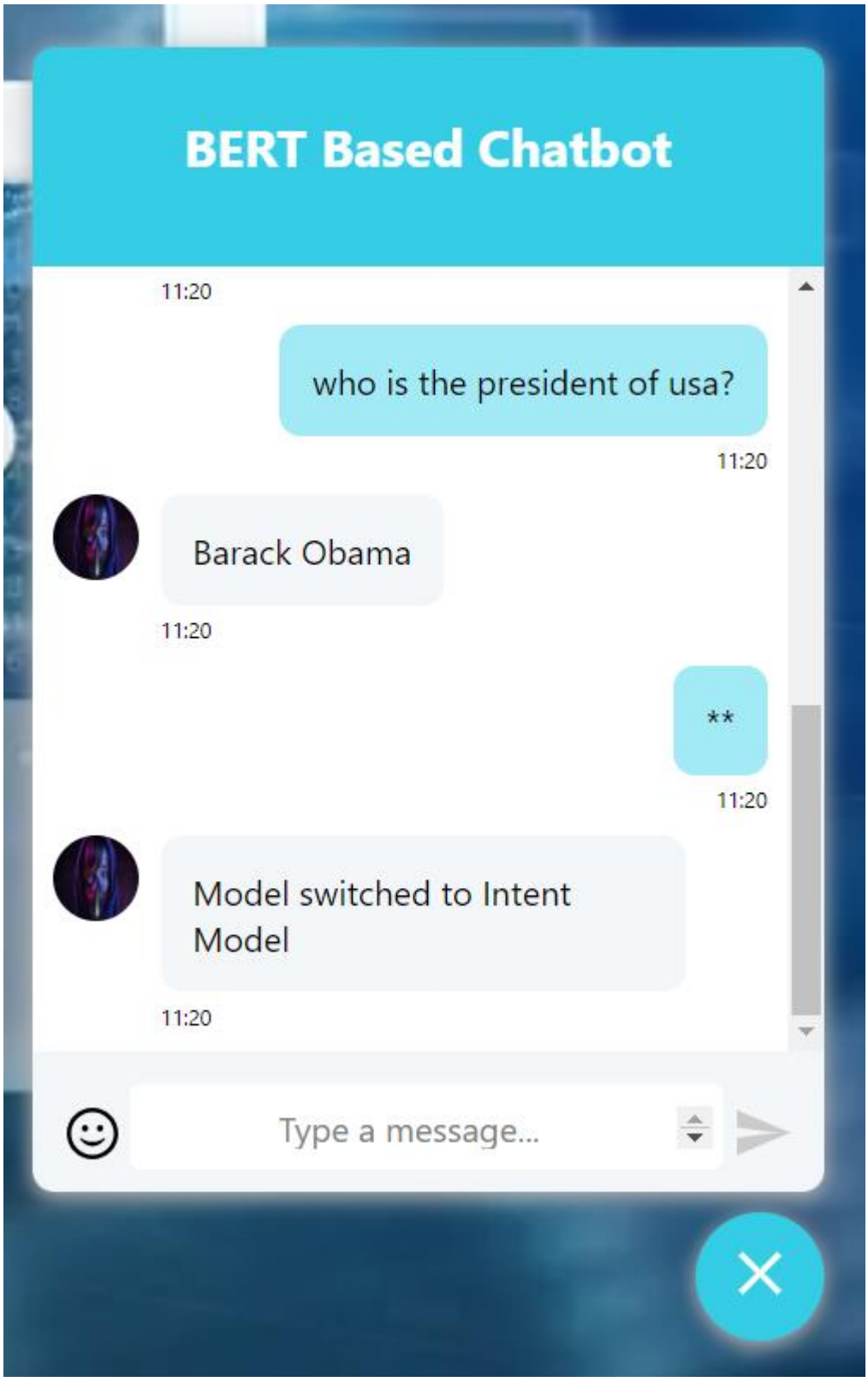

Figure 21: Switch back to Intent Model