Naan Mudhalvan

IBM project

Artificial Intelligence(Phase 3- Development)

Topic- Create a chatbot in python

By: Sneha.P.S(au411521104108)

TABELS OF CONTENTS

3.1 Dataset and its detail explanation implementation:

3.1.1 Basic Libraries:

Step 1. Install the Chatterbot and chatterbot_corpus module :tc

The first and foremost step is to install the chatterbot library. You also need to install the chatterbot_corpus library. Basically, Corpus means a bunch of words. The Chatterbot corpus contains a bunch of data that is included in the chatterbot module. The corpus is used by bots to train themselves.

Run the following pip commands on the terminal for installation:

```
pip install chatterbot
pip install chatterbot_corpus
```

Step 2. Import the modules

we have to import two classes: ChatBot from chatterbot and ListTrainer from chatterbot.trainers.

ListTrainer: Allows a chatbot to be trained using a list of strings where the list represents a conversation.

```
from chatterbot import ChatBot

from chatterbot.trainers import ListTrainer
```

Step 3. Name our Chatbot:

Now, we will give any name to the chatbot of our choice. Just create a Chatbot object. Here the chatbot is maned as “Bot” just to make it understandable.

```
bot = ChatBot('Bunny')
```

Step 4. Use of Logic Adapter:

The Logical Adapter regulates the logic behind the chatterbot that is, it picks responses for any input provided to it. This parameter contains a list of all the logical operators. When more than one logical adapter is put to use, the chatbot will calculate the confidence level, and the response with the highest calculated confidence will be returned as output.

Here we have used two logical adapters:

1. **BestMatch:** The BestMatchAdapter helps it to choose the best match from the list of responses already provided.
2. **TimeLogicAdapter:** The TimeLogicAdapter identifies statements in which a question about the current time is asked. If a matching question is detected, then a response containing the current time is returned.

```
chatbot = ChatBot(

        'BUNNY',

        logic_adapters=[

                'chatterbot.logic.BestMatch',

                'chatterbot.logic.TimeLogicAdapter'],

)
```

Step 5. Training, Communication, and Testing :

For the training process, you will need to pass in a list of statements where the order of each statement is based on its placement in a given conversation. We have to train the bot to improve its performance for this we need to call the **train()** method by passing a list of sentences. Training ensures that the bot has enough knowledge to get started with specific responses to specific inputs. After training, let’s check its communication skills. And the last step is to do testing

You have to execute the following commands now:

```
from chatterbot.trainers import ListTrainer

trainer = ListTrainer(bot)

trainer.train([

        'Hi',

        'Hello',

        'I need roadmap for Competitive Programming',

        'Just create an account on GFG and start',

        'I have a query.',

        'Please elaborate, your concern',

        'How long it will take to become expert in Coding ?',

        'It usually depends on the amount of practice.',

        'Ok Thanks',

        'No Problem! Have a Good Day!'

])
```

Now let, test the chatbot:

```
response = bunny.get_response("Good morning!")


print(response)
```

Output:

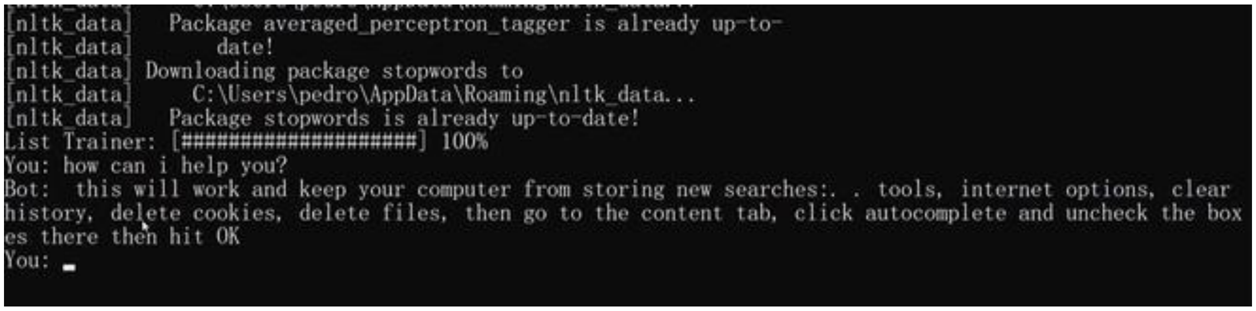
```
Good Morning
```

3.1.2 Implementation:

```
from chatterbot import ChatBot

from chatterbot.trainers import ListTrainer

from chatterbot.trainers import ListTrainer


bot = ChatBot('Bunny')


trainer = ListTrainer(bot)


trainer.train([

        'Hi',

        'Hello',

        'I need roadmap for Competitive Programming',

        'Just create an account on GFG and start',

        'I have a query.',

        'Please elaborate, your concern',

        'How long it will take to become expert in Coding ?',

        'It usually depends on the amount of practice.',

        'Ok Thanks',

        'No Problem! Have a Good Day!'

])
```

```
while True:
    request=input('you :')
    if request == 'OK' or request == 'ok':
        print('Bunny: bye')
        break
    else:
        response=bot.get_response(request)
        print('Bunny:', response)
```

Output:



```
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\pedro\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
List Trainer: [####################] 100%
You: how can i help you?
Bot:  this will work and keep your computer from storing new searches:. . tools, internet options, clear
history, delete cookies, delete files, then go to the content tab, click autocomplete and uncheck the box
es there then hit OK
You:
```

3.2 Begin building the project by load the dataset:

3.2.1 Import libraries:

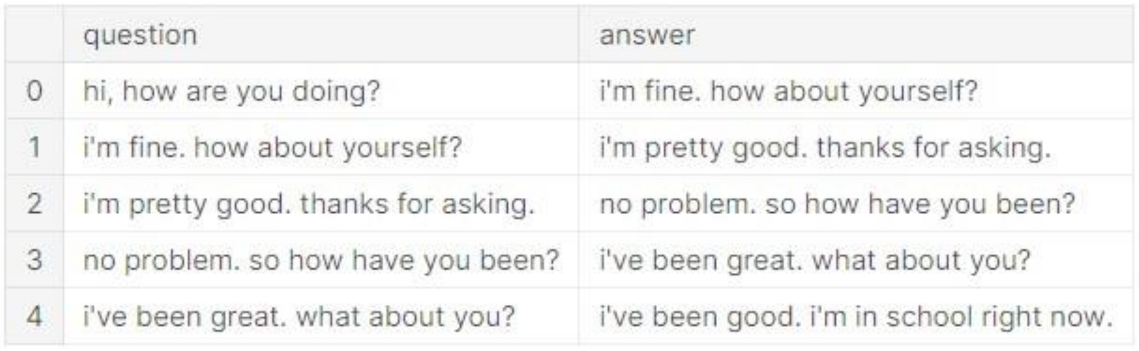
```
import tensorflow as tf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from tensorflow.keras.layers import TextVectorization

import re,string

from tensorflow.keras.layers import
LSTM,Dense,Embedding,Dropout,LayerNormalization
```

3.2.2 Dataset:

```
df=pd.read_csv('/dataset/input/simple-dialogs-for-chatbot/dialogs.txt',sep='\t',names=['qu
estion','answer'])
print(f'Dataframe size: {len(df)}')
df.head()
```

```
Dataframe size: 3725
```

Output for dataset:

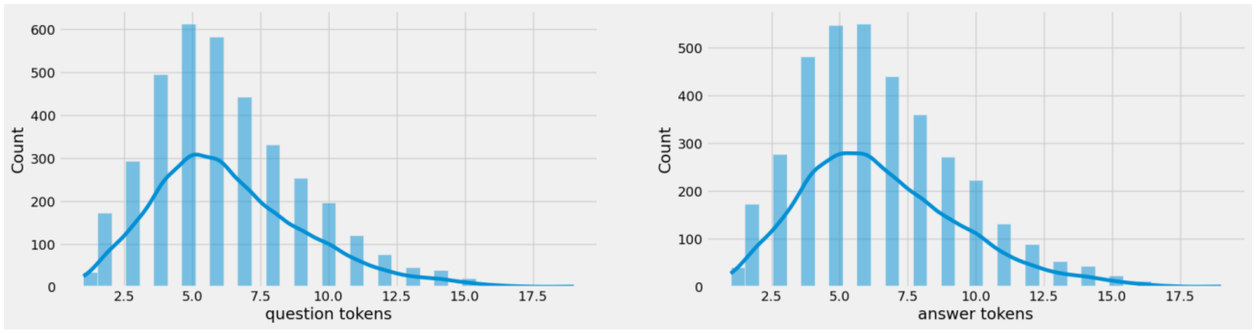


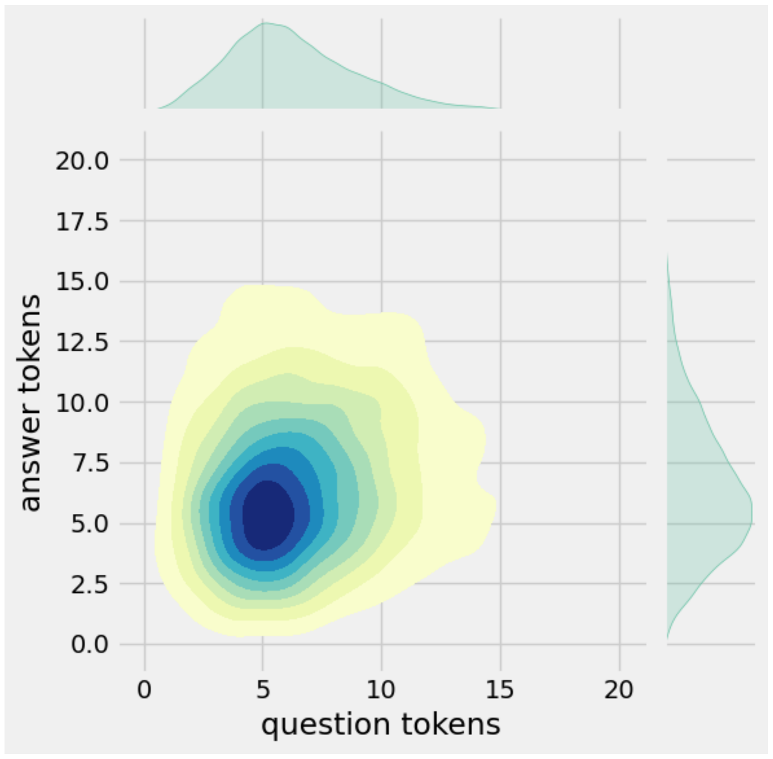
	question	answer
0	hi, how are you doing?	i'm fine. how about yourself?
1	i'm fine. how about yourself?	i'm pretty good. thanks for asking.
2	i'm pretty good. thanks for asking.	no problem. so how have you been?
3	no problem. so how have you been?	i've been great. what about you?
4	i've been great. what about you?	i've been good. i'm in school right now.

3.2.3 Data Visualization:

```
df['question tokens']=df['question'].apply(lambda x:len(x.split()))

df['answer tokens']=df['answer'].apply(lambda x:len(x.split()))
```

```
plt.style.use('fivethirtyeight')

fig,ax=plt.subplots(nrows=1,ncols=2,figsize=(20,5))

sns.set_palette('Set2')

sns.histplot(x=df['question tokens'],data=df,kde=True,ax=ax[0])

sns.histplot(x=df['answer tokens'],data=df,kde=True,ax=ax[1])

sns.jointplot(x='question tokens',y='answer
tokens',data=df,kind='kde',fill=True,cmap='YlGnBu')

plt.show()
```





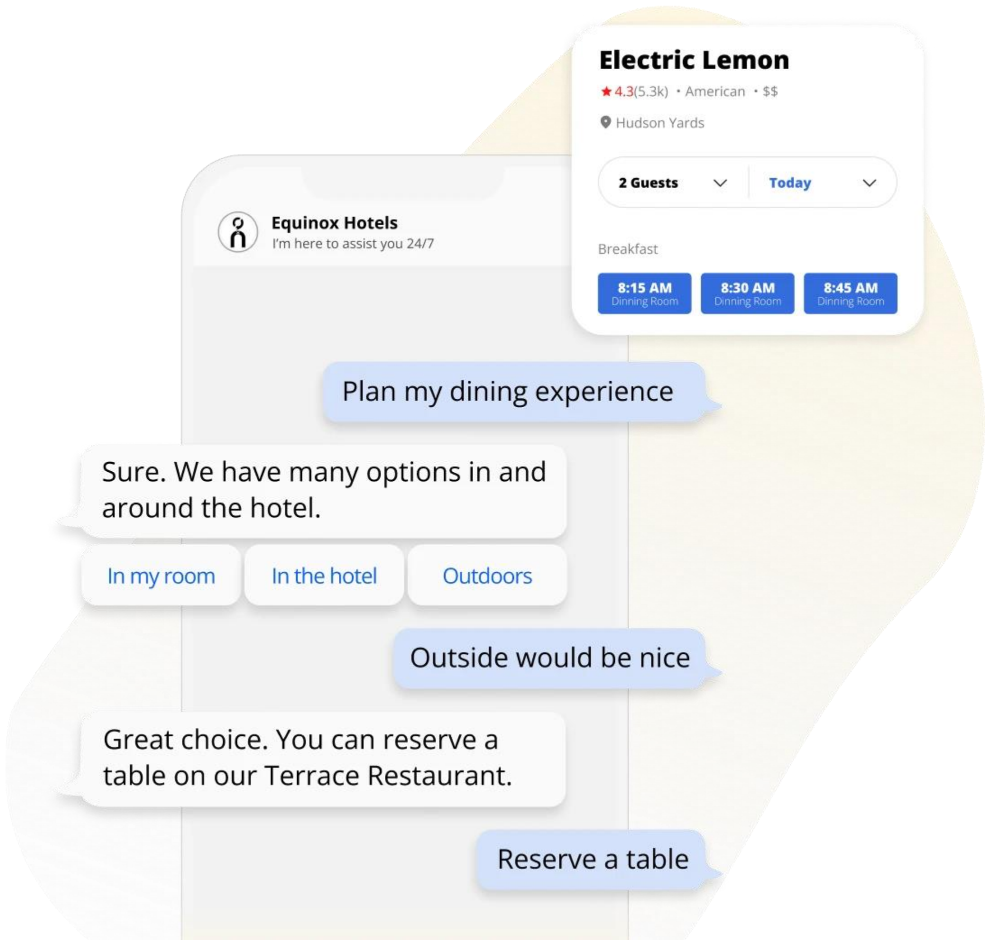
3.3 Preprocess Dataset:

3.3.1 Determine the chatbot's target purpose & capabilities

To prepare an accurate dataset, you need to know the chatbot's:

- **Purpose**: This helps in collecting relevant data and creating the conversation flow, and collecting task-oriented dialog data. For instance, a chatbot that manages customers of a restaurant might tackle conversations related to:
  - Taking orders
  - Making reservations
  - Providing the menu
  - Offering recommendations
  - Taking complaints, etc.

- **Medium**: For example, If you need a voice bot, you need completely different training data compared to the training data for a text-based bot.
- **Languages:** For example, multilingual data may need to be incorporated into the dataset.


Plan my dining experience

Sure. We have many options in and around the hotel.

Outside would be nice

Great choice. You can reserve a table on our Terrace Restaurant.

Reserve a table

3.3.2 Collect relevant data

The next step is to collect data relevant to the domain the chatbot will operate in. Data collection is one of the most important steps in preparing the dataset because that is where the data comes from. Chatbot training requires a combination of primary and secondary data, including

- Different variations of questions
- Domain-specific question-answer data
- Dialogue datasets
- Customer support data
- Transcripts of previous customer interactions
- Emails in different formats
- Social media messages
- Feedback forms

It is also important to consider the method of data collection since it can impact the quality of the dataset. Every method differs in quality, cost, and flexibility, so you need to align these factors with your project requirements. You can:

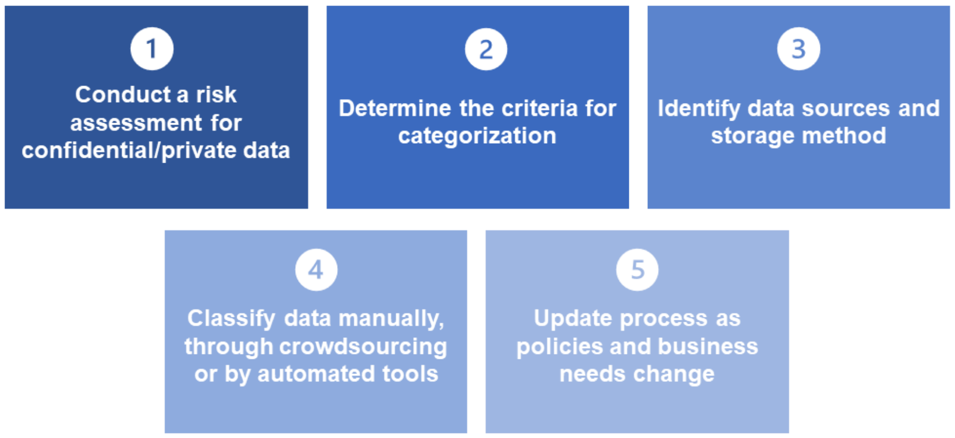
- Opt for private or in-house data collection if you can spare the budget and time and require a high level of data privacy. For instance, a patient management chatbot might work with sensitive data and, therefore, would be better suited to in-house collection of patient support datasets.
- If the chatbot requires a large amount of multilingual data, then crowdsourcing can be a suitable option. This is mainly because it offers quick access to a large pool of talent and is relatively cheaper than in-house data collection.
- You can avoid using pre-packaged or open-source datasets if quality and customization are important to your chatbot.

Its network of over 4.5 million contributors offers:

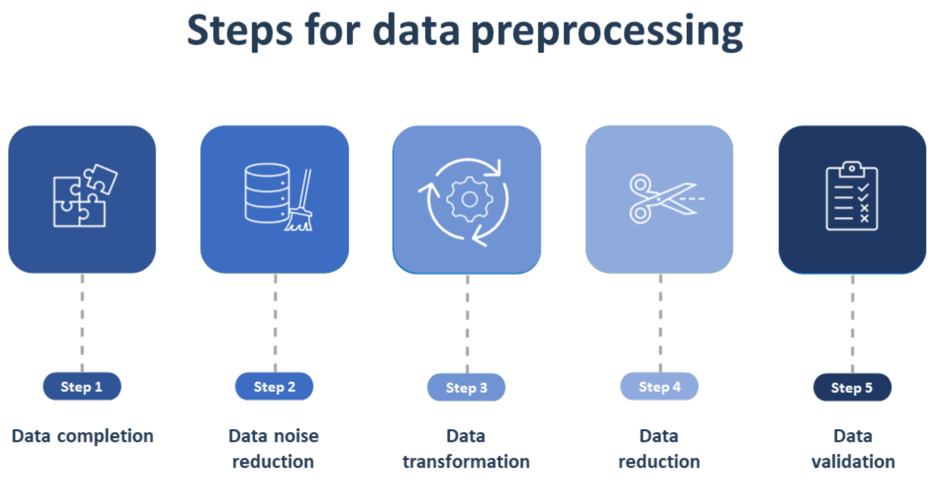
- AI data services to 4 out of 5 U.S. tech giants.
- Multilingual chatbot training datasets
- Training data for other AI/ML models
- Data transcription and image annotation services
- Data and other services for corporate or academic research

3.3.3 Categorize the data

- After gathering the data, it needs to be categorized based on topics and intents. This can either be done manually or with the help of natural language processing (NLP) tools. Data categorization helps structure the data so that it can be used to train the chatbot to recognize specific topics and intents. For example, a travel agency could categorize the data into topics like hotels, flights, car rentals, etc.
- You can consider the following 5 steps while categorizing your data:


1
Conduct a risk assessment for confidential/private data
2
Determine the criteria for categorization
3
Identify data sources and storage method
4
Classify data manually, through crowdsourcing or by automated tools
5
Update process as policies and business needs change

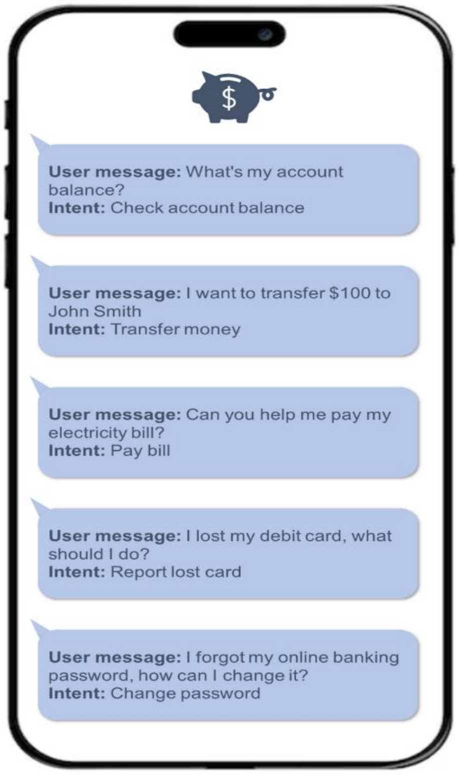
- While categorizing the data, to further improve the quality of the data, you can also preprocess it with the following 5 steps:



3.3.4 Annotate the data

After categorization, the next important step is data annotation or labeling. Labels help conversational AI models such as chatbots and virtual assistants in identifying the intent and meaning of the customer's message. This can be done manually or by using automated data labeling tools. In both cases, human annotators need to be hired to ensure a human-in-the-loop approach. For example, a bank could label data into intents like account balance, transaction history, credit card statements, etc.

Some examples of intent labels in banking:



User message: What's my account balance?
Intent: Check account balance

User message: I want to transfer $100 to John Smith
Intent: Transfer money

User message: Can you help me pay my electricity bill?
Intent: Pay bill

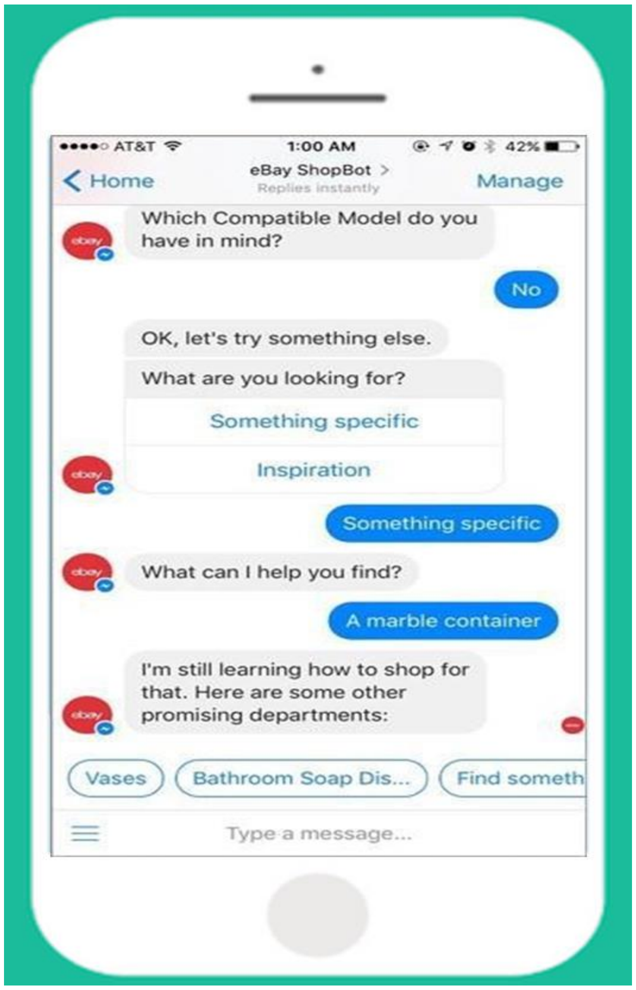
User message: I lost my debit card, what should I do?
Intent: Report lost card

User message: I forgot my online banking password, how can I change it?
Intent: Change password

3.3.5 Balance the data

To make sure that the chatbot is not biased toward specific topics or intents, the dataset should be balanced and comprehensive. The data should be representative of all the topics the chatbot will be required to cover and should enable the chatbot to respond to the maximum number of user requests.

For example, consider a chatbot working for an e-commerce business. If it is not trained to provide the measurements of a certain product, the customer would want to switch to a live agent or would leave altogether.



Which Compatible Model do you have in mind?

No

OK, let's try something else.

What are you looking for?

Something specific

Inspiration

Something specific

What can I help you find?

A marble container

I'm still learning how to shop for that. Here are some other promising departments:

Vases

Bathroom Soap Dis...

Find someth

3.3.6 Update the dataset regularly

Like any other AI-powered technology, the performance of chatbots also degrades over time. The chatbots that are present in the current market can handle much more complex conversations as compared to the ones available 5 years ago.

Chatbot training is an ongoing process. Therefore, the existing chatbot training dataset should continuously be updated with new data to improve the chatbot's performance as its performance level starts to fall. The improved data can include new customer interactions, feedback, and changes in the business's offerings.

For example, customers now want their chatbot to be more human-like and have a character. This will require fresh data with more variations of responses. Also, sometimes some terminologies become obsolete over time or become offensive. In that case, the chatbot should be trained with new data to learn those trends.

3.3.7 Test the dataset

Before using the dataset for chatbot training, it's important to test it to check the accuracy of the responses. This can be done by using a small subset of the whole dataset to train the chatbot and testing its performance on an unseen set of data. This will help in identifying any gaps or shortcomings in the dataset, which will ultimately result in a better-performing chatbot.