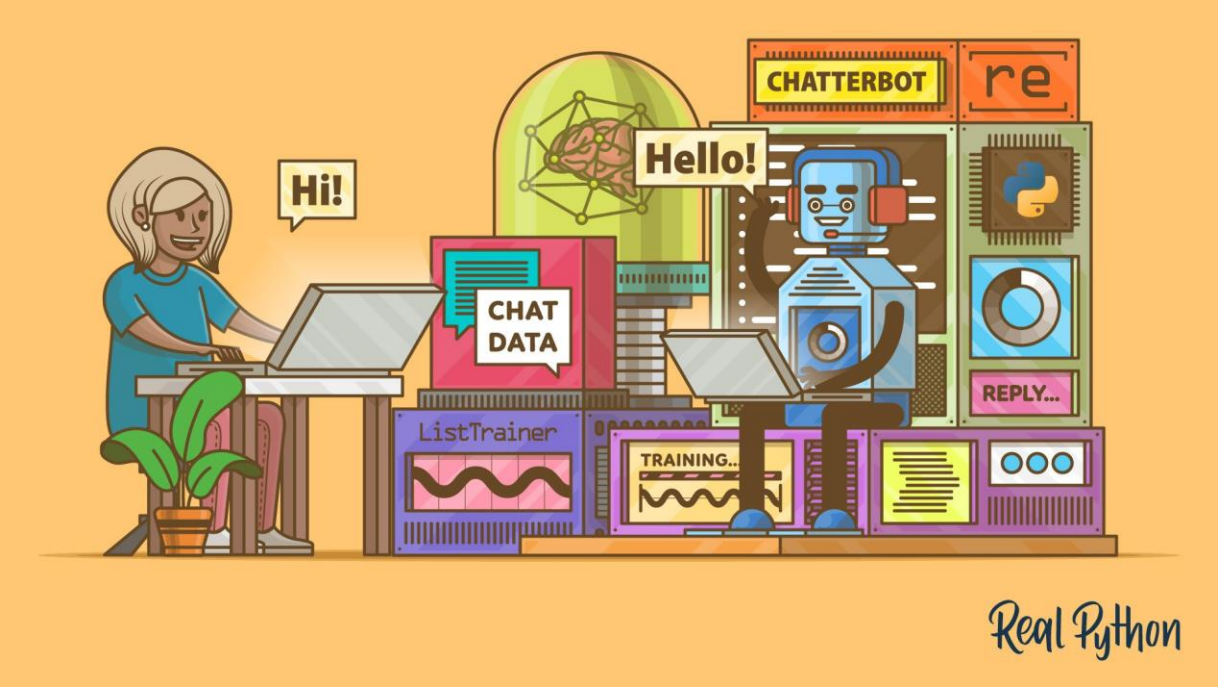
Create a chatbot in python

Phase 5 submission document

Project Title: Create a chatbot in python

Phase 5: Project Documentation & Submission

Topic: In this part you will document your project and prepare it for submission



Create a chatbot in python

Introduction:

- A chatbot is an automated software program that interacts with humans. A chatbot is merely a computer program that fundamentally simulates human conversations. A chatbot that functions through AI and machine learning has an artificial neural network inspired by the neural nodes of the human brain. Chatbots are programs that can do talk like human conversations very easily.
- For example, Facebook has a machine learning chatbot that creates a platform for companies to interact with their consumers through the Facebook Messenger application. In 2016, chatbots became too popular on Messenger. By the consequences is noted that 2016 was the entire year of chatbots.
- The software industry is mainly oriented on chatbots. Thousands of chatbots are invented on startups and used by the businesses to improve their customer service, keeping them hanging by a kind communication. According to research, nowadays chatbots are used to solve a number of business tasks across many industries like E-Commerce, Insurance, Banking, Healthcare, Finance, Legal, Telecom, Logistics, Retail, Auto, Leisure, Travel, Sports, Entertainment, Media and many others.
- Thus that was the moment to look at the chatbots as a new technology in the communication field. Nowadays various companies are using chatbots to answer quickly and efficiently some frequented asking questions from their own customers.
- Chatbot is a computer program that humans will interact with in natural spoken language and including artificial intelligence techniques such as NLP (Natural language processing) that makes the chatbot more interactive and more reliable. Based on the recent epidemiological situation, the increasing demand and reliance on electronic education has become very difficult to access to the university due to the curfew imposed, and this has led to limited access to information for academics at the university.
- This project aims to build a chatbot for Admission and Registration to answer every person who asks about the university, colleges, majors and admission policy.

Dataset Link: https://www.kaggle.com/datasets/grafstor/simple-dialogs-for-chatbot

Here's a list of tools and software commonly used in theprocess:

1. Programming Language: - Python is the most popular language for machine learning due toits extensive libraries and frameworks. You can use libraries likeNumPy,pandas, scikit-learn, and more.

2. Integrated Development Environment (IDE): - Choose an IDE for coding and running machine learningexperiments. Some popular options include Jupyter Notebook, GoogleColab, or traditional IDEs like PyCharm.

3. Machine Learning Libraries: - You'll need various machine learning libraries, including: - scikit-learn for building and evaluating machine learning models. - TensorFlow or PyTorch for deep learning, if needed. - XGBoost, LightGBM, or CatBoost for gradient boosting models.

4. Data Visualization Tools: - Tools like Matplotlib, Seaborn, or Plotly are essential for dataexploration and visualization.

5. Data Preprocessing Tools: - Libraries like pandas help with data cleaning, manipulation, andpreprocessing.

1.Design thinking and present in from of document

1. Define the Purpose: Determine the specific purpose and functionality of your chatbot. Is it for customer support, answering FAQs, or just for fun? This will guide the design and development process.
2. Choose a Framework or Library: There are several Python libraries and frameworks that can help you build a chatbot. Some popular choices include:

- NLTK (Natural Language Toolkit): For natural language processing.
- spaCy: Another NLP library that's fast and efficient.
- ChatterBot: A Python library enørifirallu fnr orea+inrt rhAthnte
- ChatterBot: A Python library specifically for creating chatbots.
- Dlalogflowor WIt.aI:These are cloud-based platforms that offer NLP capabilities for chatbot development.

3. Data Collection: Gather and preprocess the data your chatbot will use to understand and respond to user input. This may include a corpus of text, FAQs, or other relevant information.

4. NLP Processing: use your chosen NLP library or framework to process and understand user input. This involves tasks like tokenization, part-ofspeech tagging, and entity recognition.

5. Build a Dialogue Engine: Create a dialogue engine that can generate responses based on user input and the data you've collected. This can be rule-based, machine learning-based, or a combination of both.

6. user Interface (UI): Decide on the user interface for your chatbot. It could be a web-based chat widget, a command-line interface, or integrated into an existing application.

7. Integration: If your chatbot needs to interact with external systems or APIs, implement the necessary integrations.

8. Testing and Training: Thoroug hlytest your chatbot and provide training data to improve its accuracy and response quality. Iterate on the model and dialogue engine as needed. Needed.

9. Deployment: Deploy your chatbot to a server or platform of your choice. Ensure it's accessible to users.

10. Monitoring and Maintenance: Continuously monitor your chatbot's performance, gather user feedback, and make improvements over time. This may involve updating the modeli adding more data, or enhancing the dialogue engine,

11. Security and Privacy: Implement security measures to protect user data and privacy, especially if your chatbot handles sensitive information.

12. Scaling: If your chatbot gains popularity, be prepared to scale your infrastructure to handle increased traffic and usage.

2.**Design into innovation:**

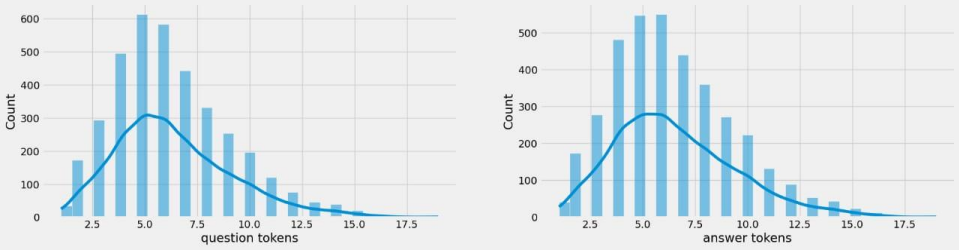
- Machine Learning Integration: Implement machine learning algorithms for better natural language processing, sentiment analysis, or user intent recognition.
- Multimodal Capabilities: Enable the bot to comprehend and respond using not just text but also images, voice, or even video.
- Personalization: Integrate personalization features to tailor responses based on user history or preferences, creating a more personalized interaction.

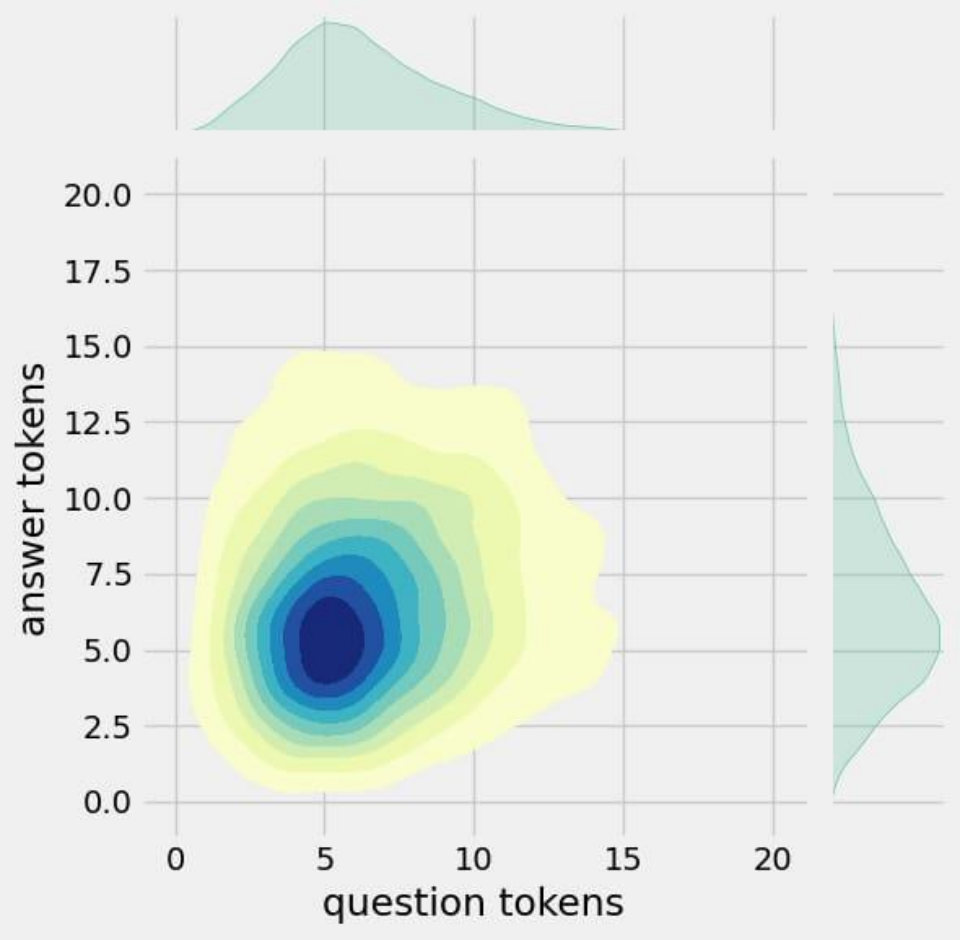
```
import tensorflow as tf

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
```

Data visualization:

```
t.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['answ 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=pl er 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.Build loading and preprocessing the dataset:

1. Data Loading: First, you need to load your dataset. This could be from a CSV file, a database, or any other source. In Python, you might use libraries like pandas for this task.

```
import pandas as pd

dataset = pd.read_csv('your_dataset.csv(‘your_dataset.csv’)
```

2. Data Exploration: After loading the data, it's a good practice to explore it. Check for the number of rows and columns, data types, and basic statistics.

```
print(dataset.shape)

print(dataset.info())

print(dataset.describe())
```

3. Handling Missing Data: You may need to handle missing data by filling in missing values or removing rows with missing values.

Dataset.dropna() # To remove rows with missing values

dataset.fillna(value) # To fill in missing values

4. Data Preprocessing: This step involves transforming the data as needed. Common preprocessing steps include encoding categorical variables, scaling numerical features, and feature engineering.

from sklearn.preprocessing import LabelEncoder, StandardScaler label_encoder = LabelEncoder() dataset['categorical_column'] = label_encoder.fit_transform(dataset['categorical_column']) scaler = StandardScaler() dataset['numeric_column'] = scaler.fit_transform(dataset['numeric_column'].values.reshape(- 1, 1))

5. Data Splitting: If you're working on a supervised learning task, split the dataset into training and testing sets.

from sklearn.model_selection import train_test_split X = dataset.drop('target_column' , axis= 1) y = dataset['target_column']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size= 0.2, random_state= 42)

6. Feature Selection: Choose relevant features for your model if necessary.

7. Data Augmentation (for certain tasks like image processing).

8. Data Normalization: Normalize the data if needed.

9. Data Loading and Augmentation: If you're working with deep learning, you might use libraries like TensorFlow or PyTorch to load and augment your data.

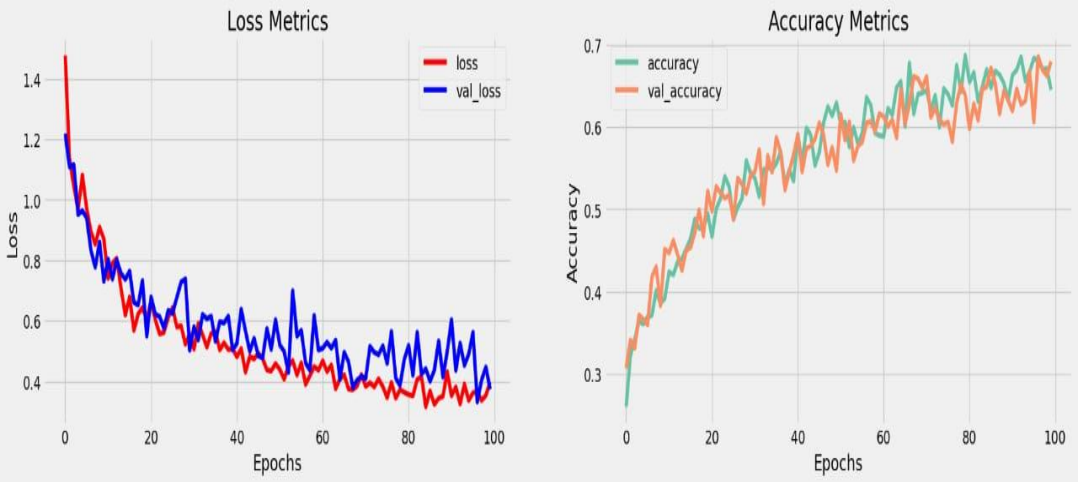
TOKENIZATION:

```
def tokenize(lang):
    lang_tokenizer = tf.keras.preprocessing.text.Tokenizer(
      filters='')

    #build vocabulary on unique words
    lang_tokenizer.fit_on_texts(lang)
```

```
    return lang_tokenizer
```

Visualize Metrics:

```
 fig,ax=plt.subplots(nrows=1,ncols=2,figsize=(20,5))
ax[0].plot(history.history['loss'],label='loss',c='red')
ax[0].plot(history.history['val_loss'],label='val_loss',c = 'blue')
ax[0].set_xlabel('Epochs') ax[1].set_xlabel('Epochs')
ax[0].set_ylabel('Loss') ax[1].set_ylabel('Accuracy')
ax[0].set_title('Loss Metrics') ax[1].set_title('Accuracy Metrics')
ax[1].plot(history.history['accuracy'],label='accuracy')
ax[1].plot(history.history['val_accuracy'],label='val_accuracy')
ax[0].legend() ax[1].legend() plt.show()
```



4.continue building the chatbot by integrating it into a web app using flask:

Feature Selection Techniques:

Mutual Information:

Mutual information Calculate between features and the target variable (e.g., intent or response) and select the most informative features.

Chi-Square Test:

Use chi-square statistical test to select relevant features for classification tasks.

Feature Importance from Models:

Train a model (e.g., Random Forest or XGBoost) and use feature importance scores to select features.

Recursive Featu Elimination (RFE):

Iteratively remove the least important features based on a model's performance.

Correlation:

Analyze feature-target correlations and remove features with low correlation.

Dimensionality Reduction:

Consider techniques like Principal Component Analysis (PCA) to reduce dimensionality while preserving information.

Feature Scaling:

Scale or normalize the selected features to ensure they have a consistent range.

Model Building:

Train your chatbot model (e.g., a sequence-tosequence model, intent classification, or dialogue management) using the selected features.

Evaluation:

Evaluate your chatbot's performance using appropriate metrics (e.g., accuracy, F1-score, or BLEU score for language generation).

Iterate:

Depending on the performance, you may need to iterate through feature selection and model tuning.

Here's a basic Python code snippet to demonstrate feature extraction and selection using TF-IDF and mutual information:

from sklearn.feature_extraction.text import TfidfVectorizer from sklearn.feature_selection import SelectKBest, mutual_info_classif

Assuming you have a list of text messages (X) and their corresponding labels (y) tfidf_vectorizer = TfidfVectorizer() X_tfidf = tfidf_vectorizer.fit_transform(X)

Use mutual information to select the top 'k' features k_best = SelectKBest(score_func=mutual_info_classif, k=100) # Adjust 'k' as needed X_selected = k_best.fit_transform(X_tfidf, y)

Model training:
model=ChatBotTrainer(encoder,decoder,name='chatbot_trainer ') model.compile(loss=tf.keras.losses.SparseCategoricalCrossentropy(), optimizer=tf.keras.optimizers.Adam(learning_rate=learning_rate), weighted_metrics=['loss','accuracy']) model(_[:2])

output:

```
<tf.Tensor: shape=(149, 30, 2443), dtype=float32, numpy=
array([[[3.40592262e-04, 5.73484940e-05, 2.12948853e-05, ...,
         7.20679745e-05, 1.54536311e-03, 2.35993255e-04],
        [1.46621116e-03, 8.02504110e-06, 5.40619949e-05, ...,
         1.91874733e-05, 9.72440175e-05, 7.64339056e-05],
        [9.69291723e-05, 2.74417835e-05, 1.37613132e-03, ...,
         3.60095728e-05, 1.55378671e-04, 1.83973272e-04],
        ...,
        [1.90027885e-03, 6.92659756e-04, 1.43461803e-04, ...,
         1.95525470e-04, 1.71066222e-05, 1.02524005e-04],
        [1.90027885e-03, 6.92659756e-04, 1.43461803e-04, ...,
         1.95525470e-04, 1.71066222e-05, 1.02524005e-04],
        [1.90027885e-03, 6.92659756e-04, 1.43461803e-04, ...,
         1.95525470e-04, 1.71066222e-05, 1.02524005e-04]],

       [[9.24730921e-05, 3.46553512e-04, 2.07866033e-05, ...,
         3.65934626e-04, 7.63039337e-04, 5.52638434e-04],
        [8.46863186e-05, 3.65541164e-05, 2.54740953e-05, ...,
         7.12379551e-05, 3.62201303e-04, 4.16714087e-04],
        [2.30146630e-04, 3.91469621e-06, 2.72463716e-04, ...,
         9.26126595e-05, 1.03836363e-04, 1.40792166e-04],
        ...,
        [6.84961735e-04, 9.07644513e-04, 2.86691647e-04, ...,
         3.87946144e-04, 6.09236558e-05, 1.12995331e-05],
        [6.84961735e-04, 9.07644513e-04, 2.86691647e-04, ...,
         3.87946144e-04, 6.09236558e-05, 1.12995331e-05],
        [6.84961735e-04, 9.07644513e-04, 2.86691647e-04, ...,
         3.87946144e-04, 6.09236558e-05, 1.12995322e-05]],

       [[1.19036995e-03, 8.10516722e-05, 2.42324077e-05, ...,
         4.99442758e-05, 6.67208573e-04, 9.55566764e-04],
        [1.53046989e-04, 9.76863957e-05, 4.96972689e-06, ...,
         3.24743196e-05, 2.12563842e-04, 1.18708890e-03],
        [9.40205529e-04, 1.80782794e-04, 7.26205144e-06, ...,
         1.96355060e-04, 8.16940737e-05, 1.38416886e-03],
        ...,
        [3.52622545e-03, 1.26781175e-03, 1.02695449e-04, ...,
         2.35450850e-03, 3.25187625e-06, 9.46984728e-05],
```

```
 [3.52622545e-03, 1.26781175e-03, 1.02695449e-04, ...,
  2.35450850e-03, 3.25187625e-06, 9.46984728e-05],
 [3.52622545e-03, 1.26781175e-03, 1.02695449e-04, ...,
  2.35450850e-03, 3.25187625e-06, 9.46984728e-05]],

...,

[[9.03617911e-05, 1.57651404e-04, 1.02747028e-04, ...,
  2.20922651e-04, 3.61504179e-04, 2.32456136e-03],
 [1.55469708e-04, 1.53608169e-04, 1.14945491e-04, ...,
  1.88878359e-04, 5.11967926e-04, 5.13108505e-04],
 [8.27641197e-05, 2.83437112e-05, 6.29429938e-04, ...,
  2.15980137e-04, 3.02832137e-04, 1.77760507e-04],
 ...,
 [2.41102395e-03, 1.29279669e-03, 9.11735406e-05, ...,
  4.06600971e-04, 7.58682154e-06, 6.05909081e-05],
 [2.41102395e-03, 1.29279669e-03, 9.11735406e-05, ...,
  4.06600971e-04, 7.58682154e-06, 6.05909081e-05],
 [2.41102395e-03, 1.29279669e-03, 9.11735406e-05, ...,
  4.06600971e-04, 7.58682154e-06, 6.05909081e-05]],

[[3.99837241e-04, 2.36026899e-05, 6.89777007e-05, ...,
  5.94239136e-05, 4.32556757e-04, 4.60232928e-04],
 [3.88111075e-04, 8.31133584e-05, 1.11861555e-04, ...,
  3.03280340e-05, 2.54765386e-04, 2.82170397e-04],
 [2.12516752e-03, 7.19837190e-05, 1.88700986e-04, ...,
  1.86366087e-04, 7.02239413e-05, 2.54370330e-04],
 ...,
 [4.56329063e-03, 2.23812275e-03, 2.37343236e-04, ...,
  2.64523784e-04, 4.05454011e-05, 1.55662783e-04],
 [4.56329063e-03, 2.23812275e-03, 2.37343236e-04, ...,
  2.64523784e-04, 4.05454011e-05, 1.55662783e-04],
 [4.56329063e-03, 2.23812275e-03, 2.37343236e-04, ...,
  2.64523784e-04, 4.05454011e-05, 1.55662783e-04]],

[[3.24600202e-04, 9.31067043e-05, 4.60048941e-05, ...,
  6.66230699e-05, 5.76460850e-04, 1.52416309e-04],
 [7.51478728e-05, 7.63997741e-05, 2.09082973e-05, ...,
  2.55555002e-04, 2.28998848e-04, 4.37303359e-04],
 [1.03114333e-04, 1.55743372e-04, 9.97955431e-06, ...,
  1.12485175e-03, 4.80950950e-03, 6.83143327e-04],
 ...,
 [5.20280097e-03, 3.23211338e-04, 2.47709468e-05, ...,
```

```
         3.07609705e-04, 6.09844255e-06, 8.61325825e-05],
        [5.20280097e-03, 3.23211338e-04, 2.47709468e-05, ...,
         3.07609705e-04, 6.09844255e-06, 8.61325825e-05],
        [5.20280097e-03, 3.23211338e-04, 2.47709468e-05, ...,
         3.07609705e-04, 6.09844255e-06, 8.61325825e-05]]],
dtype=float32)>
```

Model evaluation:

Evaluating a chatbot model in Python typically involves assessing its performance using various metrics and techniques. Here's a basic outline of the steps you can follow:

Choos e Evaluation Metrics:

Select appropriate metrics to evaluate the chatbot's performance, such as:

 BLEU score for measuring the quality of generated responses.
 Perplexity to assess the model's language generation capabilities.

 F1 score, accuracy, or precision/recall for intent recognition in task-oriented chatbots.

 Human evaluation with annotators to assess response quality.

Testing and Evaluation:

 Generate responses using the chatbot model on the test dataset.

 Calculate the chosen evaluation metrics to measure its performance.

 For human evaluation, gather feedback from human annotators on the chatbot's responses. Here's a simple example using the NLTK library in Python to calculate BLEU scores for generated responses:

from nltk.translate.bleu_score import sentence_bleu

Reference and candidate responses reference = ["the quick brown fox jumps over the lazy dog"]

candidate = "a quick brown fox jumps over a lazy dog"

Calculate BLEU score score = sentence_bleu([reference], candidate) print(f"BLEU Score: {score}")

Future engineering:

The field of engineering in chatbots using Python is likely to continue evolving in the future. Some potential advancements may include:

Natural Language Understanding:

Improved methods for chatbots to understand and interpret human language more accurately and contextually.

Emotional Intelligence:

Developing chatbots that can detect and respond to human emotions, making interactions more empathetic and personalized.

Multimodal Capabilities:

Integration of text, speech, and visual inputs/outputs to create more versatile and humanlike chatbots.

Reinforcement Learning:

Utilizing reinforcement learning techniques to make chatbots more adaptable and capable of learning from interactions.

 Better NLP Models: Advancements in Natural Language Processing (NLP) models, like GPT-4 or successors, that can generate more coherent and context-aware responses.

 Security and Ethics: Enhanced security measures and ethical guidelines to address issues related to data privacy, bias, and misuse of chatbots.

 Industry-Specific Applications: Customized chatbots for various industries, such as healthcare, finance, and customer service, to provide specialized assistance.

Interperability : Improved interoperability with other systems and platforms, enabling seamless integration of chatbots into various applications.

 Self-improvement: Chatbots that can autonomously learn and improve over time with minimal human intervention.

 Voice Assistants: Integration of chatbots into voice assistants like Siri or Alexa, making them more conversational and helpful.

ADVANTAGES:

- Flexibility: Python provides various libraries and frameworks (like NLTK, TensorFlow, PyTorch, etc.) for natural language processing, making it flexible for chatbot development.
- Easy to Learn and Use: Python's simplicity and readability make it an excellent choice for both beginners and experienced developers.
- Vast Libraries: The Python ecosystem offers numerous libraries that streamline chatbot development, including pre-built frameworks such as Rasa or ChatterBot.

- Community Support: Python has a large and active community, providing ample resources, tutorials, and support for developing chatbots.

- Integration: Python easily integrates with other technologies, APIs, and platforms, allowing chatbots to interact with databases, web services, and more.

- Scalability: With various deployment options and scalability potential, Python chatbots can cater to small-scale or enterprise-level requirements.

These advantages make Python a popular choice for developing chatbots due to its ease of use, vast resources, and powerful libraries.

DISADVANTAGES:

- ★ Performance: Depending on the complexity, Python might not be as fast as other languages, leading to potential performance issues in high-traffic scenarios.
- ★ Resource Intensive: Python might consume more system resources compared to other languages, impacting scalability and efficiency.

- ★ Language Support: While Python has various libraries, it might have limitations in terms of language processing, especially in handling certain languages or specific dialects.

- ★ Training Data Size: Python chatbots could face constraints in handling massive datasets or training models due to memory limitations.

- ★ Maintenance: Python chatbots might require frequent updates due to library version changes, potentially impacting the bot's stability.

- ★ Limited Development Frameworks: There might be fewer specialized chatbot frameworks or tools available in Python compared to other languages.

- ⋆ Natural Language Understanding (NLU): Achieving high accuracy in NLU might be challenging in Python due to limitations in available NLP libraries.

However, these limitations can often be mitigated or worked around by optimizing code, using efficient libraries, and implementing strategies tailored to the specific use case of the chatbot.

BENEFITS:

- ♦ Ease of Development: Python provides a simple and readable syntax, making it easy to develop, maintain, and iterate on chatbot projects.
- ♦ Abundant Libraries: Python boasts rich libraries and frameworks like NLTK, spaCy, and TensorFlow, enabling natural language processing (NLP) and machine learning capabilities for chatbot development.
- ♦ Scalability: Python's versatility allows for scalable bot development, enabling integration with various platforms and systems.
- ♦ Community Support: Python has a vast and supportive community, offering numerous resources, tutorials, and pre-existing code to expedite bot development.
- ♦ Integration Flexibility: Python's versatility permits seamless integration with web services, APIs, databases, and various platforms, enhancing a chatbot's functionality and utility.
- ♦ Machine Learning Capabilities: Python's extensive machine learning libraries allow for the development of more

sophisticated chatbots capable of learning and adapting to user interactions over time.

- Cost-Effectiveness: Python's open-source nature and vast resources contribute to cost-effective chatbot development, particularly for startups or businesses with limited resources.

CONCLUSION:

- In this project, we have introduced a chatbot that is able to interact with users. This chatbot can answer queries in the textual user input. For this purpose, AIML with program-o has been used.
- The chatbot can answer only those questions which he has the answer in its AIML dataset. So, to increase the knowledge of the chatbot, we can add the APIs of Wikipedia, Weather Forecasting Department, Sports, News, Government and a lot more.
- In such cases, the user will be able to talk and interact with the chatbot in any kind of domain. Using APIs like Weather, Sports, News and Government Services, the chatbot will be able to answer the questions outside of its dataset and which are currently happening in the real world.
- The next step towards building chatbots involves helping people to facilitate their work and interact with computers using natural language or using their set of rules. Future Such chatbots, backed by machine-learning technology, will be able to remember past conversations and learn from them to answer new ones.
- The challenge would be conversing with the various multiple bot users and multiple users. As future work, we

can make a chatbot that is based on AIML and LSA. This technology will enable a client to interact with a chatbot in a more natural fashion.

- We can enhance the discussion by including and changing patterns and templates for general client queries using AIML and the right response are given more often than LSA.