PROJECT TITLE: CREATE A CHATBOT IN PYTHON

PHASE 2: INNOVATION

INTRODUCTION:

The innovation phase of this projects revolves around the collection of data and processing it. It also describes respective machine learning algorithms for the chatbot model upon which the bot is being trained. In order to train the model or the chatbot, available dataset from Kaggle has been used. Let's explore from data processing to modelling one after other.

LIBRARIES USED:

- *TENSORFLOW:*
  TensorFlow is a popular open-source machine learning framework developed by Google. While it may not be the primary choice for building chatbots compared to libraries like PyTorch, it still plays important roles in chatbot development, particularly when it comes to implementing deep learning models and components.it also provides deep learning framework and model training.Tensorflow helps with customization of chatbot and can be used for implementing NLP components.

- *TRANSFORMER:*
  The "transformers" library, developed by Hugging Face, is a powerful library that plays a significant role in chatbot development, especially for chatbots that use transformer-based models. Transformers have revolutionized natural language processing (NLP) and chatbot technology due to their ability to handle contextual information effectively.

- *TORCH:*
  The torch library, which is part of PyTorch, plays a crucial role in many aspects of building a chatbot, especially when you are implementing deep learning-based chatbots. PyTorch is a popular deep learning framework in Python, and it offers several features and functionalities that are essential for creating chatbots**.**

- *GPT3:*
  GPT-3 (Generative Pre-trained Transformer 3) plays a significant role in chatbot development, offering several advantages and capabilities when integrated into a chatbot system.

- *KERAS:*
  Keras is a high-level deep learning framework in Python that can be an important component in the development of chatbots, especially when chatbots involve machine learning models.

- *LSTM (Long Short-Term Memory):*
  Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) architecture that plays a significant role in chatbot development, particularly in natural language processing and sequence modelling.

- *NUMPY:*
  NumPy is a fundamental library in Python for numerical computations and data manipulation. While it may not be directly involved in chatbot interactions, it plays a crucial role in chatbot development, particularly in handling and processing numerical and array-based data.

- *PANDAS:*
  Pandas is a widely-used Python library for data manipulation and analysis. While it may not be directly involved in chatbot interactions, it can play several important roles in chatbot development, particularly in handling and processing data.

- *SEABORN:*
  Seaborn is a data visualization library in Python that is built on top of Matplotlib. While it's not directly involved in chatbot interactions, Seaborn can play a valuable role in chatbot development for data visualization and analysis.

- *MATPLOTLIB:*
  Matplotlib is a popular data visualization library in Python that can serve several roles in chatbot development, especially when it comes to visualizing data and results for users.

- *SKLEARN:*
  Scikit-learn, often abbreviated as sklearn, is a versatile machine learning library in Python. It plays a vital role in chatbot development, particularly when chatbots require machine learning models for various tasks.

DATA PROCESSING:

DATA CLEANSING:

Data cleansing, also known as data cleaning or data preprocessing, is a critical step in chatbot development. It involves the process of cleaning and transforming the raw data used for training or fine-tuning a chatbot to ensure that it is of high quality, consistent, and suitable for the chatbot's intended purpose. Here are some key steps and techniques for data cleansing for chatbot development:

1. **Data Collection**: Start by collecting the necessary data for your chatbot. This data may include user queries, responses, intent labels, entity recognition data, and any other information relevant to your chatbot's domain.

   code:

```
import nltk
import string
import pandas as pd
import nlp_utils as nu
import matplotlib.pyplot as plt
df=pd.read_csv('/content/dialogs.txt',sep='\t',names=['Question','Answer'])
```

```
[ ]  print(df)
```

   Output:

```
                                          Question  \
0                                hi, how are you doing?
1                          i'm fine. how about yourself?
2                   i'm pretty good. thanks for asking.
3                     no problem. so how have you been?
4                      i've been great. what about you?
...                                                 ...
3720   that's a good question. maybe it's not old age.
3721                              are you right-handed?
3722                                  yes. all my life.
3723  you're wearing out your right hand. stop using...
3724        but i do all my writing with my right hand.

                                             Answer
0                          i'm fine. how about yourself?
1                   i'm pretty good. thanks for asking.
2                     no problem. so how have you been?
3                      i've been great. what about you?
4              i've been good. i'm in school right now.
...                                                 ...
3720                              are you right-handed?
3721                                  yes. all my life.
3722  you're wearing out your right hand. stop using...
3723        but i do all my writing with my right hand.
3724  start typing instead. that way your left hand ...

[3725 rows x 2 columns]
```

2. **Data Inspection**: Examine the collected data to identify issues and inconsistencies. Common problems include spelling errors, missing values, duplicate entries, and irrelevant or noisy data.

3. **Text Preprocessing**: Text data, such as user queries and responses, often requires preprocessing. This involves tasks like:

Code:

```
[ ]  # Removing special characters
     import re
     # importing regular expressions
     punc_lower = lambda x: re.sub('[%s]' % re.escape(string.punctuation), ' ', x.lower())
     # Lower case conversion
     remove_n = lambda x: re.sub("\n", " ", x)
     # removing \n and replacing them with empty value
     remove_non_ascii = lambda x: re.sub(r'[^\x00-\x7f]',r' ', x)
     # removing non ascii characters
     alphanumeric = lambda x: re.sub('\w*\d\w*', ' ', x)
     # removing alpha numeric values
     df['Question'] = df['Question'].map(alphanumeric).map(punc_lower).map(remove_n).map(remove_non_ascii)
     # using map function and applying the function on query column
     df['Answer'] = df['Answer'].map(alphanumeric).map(punc_lower).map(remove_n).map(remove_non_ascii)
     # using map function and applying the function on response column
     print(df)
     # final cleaned dataset
```

Output:

```
                                          Question  \
0                               hi  how are you doing
1                         i m fine  how about yourself
2                  i m pretty good  thanks for asking
3                    no problem  so how have you been
4                     i ve been great  what about you
...                                               ...
3720   that s a good question  maybe it s not old age
3721                             are you right handed
3722                                 yes  all my life
3723  you re wearing out your right hand  stop using...
3724       but i do all my writing with my right hand

                                           Answer
0                         i m fine  how about yourself
1                  i m pretty good  thanks for asking
2                    no problem  so how have you been
3                     i ve been great  what about you
4             i ve been good  i m in school right now
...                                               ...
3720                             are you right handed
3721                                 yes  all my life
3722  you re wearing out your right hand  stop using...
3723       but i do all my writing with my right hand
3724  start typing instead  that way your left hand ...

[3725 rows x 2 columns]
```

- **Lowercasing**: Convert all text to lowercase to ensure case-insensitive matching.
- **Tokenization**: Split text into individual words or tokens for analysis.
- **Stop Word Removal**: Remove common words like "and," "the," and "in" that don't carry much meaning.
- **Special Character Removal**: Eliminate punctuation and special characters.

- **Stemming or Lemmatization**: Reduce words to their root forms to handle variations.
- **Data Deduplication:** Remove duplicate or near-duplicate entries from the data, which can distort the chatbot's learning process.
- **Entity Recognition:** If your chatbot needs to recognize specific entities (e.g., names, dates, locations), you can use entity recognition tools or custom patterns to identify and label these entities in the data.
- **Data Labeling:** For supervised learning tasks like intent recognition, ensure that the data is properly labeled with the correct intent or entity labels. Consistently labeled data is crucial for training accurate models.
- **Data Splitting**: After cleansing, split the data into training, validation, and test sets. This allows you to train, validate, and evaluate your chatbot models effectively.

NLP PROCESSING (Natural Language Processing):

NLP is a field of computer science that deals with the interaction between computers and human language. NLP algorithms and models are used to enable chatbots to understand and generate human-like language. Common NLP platforms for chatbots include Dialog flow, Rasa, and Amazon Lex. A marketing agency could use generative NLP processing to generate creative marketing copy. Healthcare website could use emotionally intelligent NLP processing to provide support to patients who are feeling anxious or stressed.

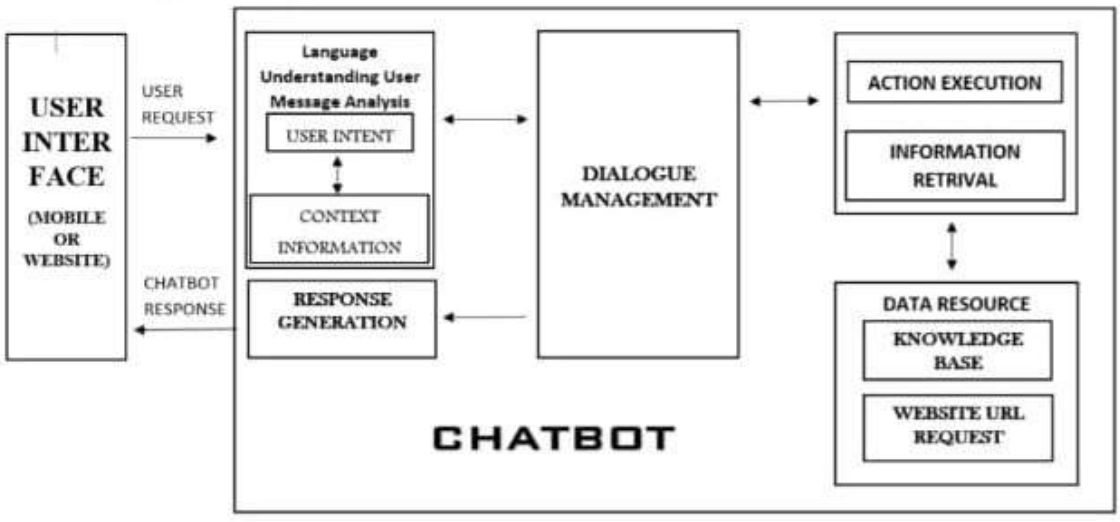
The basic text processing in NLP:

- *Sentence Segmentation:*
  This is a fundamental step in many NLP applications because it helps in breaking down the text into smaller units for analysis, processing, and further understanding. Sentence segmentation can be a non-trivial task because sentences in natural language text can have various structures, making it challenging to determine where one sentence ends and the next begins.
- *Normalization:*
  The goal of normalization is to reduce text variations, making it easier to process, analyze, and compare textual information. Normalization is essential in many NLP tasks to ensure that the text is consistent and uniform, as it can significantly impact the performance of algorithms and models.

- *Tokenization:*
  Tokenization is a fundamental natural language processing (NLP) task that involves breaking down a text or document into smaller units called tokens. These tokens are typically words, phrases, or symbols, and the goal of tokenization is to facilitate the analysis and processing of text data by splitting it into manageable units.

- *Stop word Removal:*
  Stop word removal is a common text pre-processing technique in natural language processing (NLP) that involves eliminating words that are considered common and non-informative from a text or document. These "stop words" are often excluded from analysis because they do not contribute significantly to the meaning of the text and may add noise to NLP tasks. The list of stop words typically includes common words such as "the," "and," "is," "in," "to," and other frequently occurring words.
- *Lemmatization and Stemming:*
  Lemmatization and stemming are text normalization techniques used in natural language processing (NLP) to reduce words to their base or root forms. The primary goal of both methods is to standardize text data, making it easier to analyze, process, and compare. However, they differ in their approaches and outcomes.

NLP PROCESSING



CHATBOT'S LEARNING ALGORITHMS:

ENSEMBLING TECHNIQUE:

Ensuring the accuracy and robustness of chatbot responses is crucial for delivering a seamless user experience. Ensemble techniques can be employed to enhance chatbot performance.

Ensemble techniques involve combining the predictions or responses from multiple models to improve overall accuracy and robustness. In the context of chatbots, this means creating a diverse set of models, each with its unique approach and strengths.

```
# Initialize GPT-2 model and tokenizer
model_name = "gpt2"
tokenizer = GPT2Tokenizer.from_pretrained(model_name)
model = GPT2LMHeadModel.from_pretrained(model_name)

# Fine-tune the GPT-2 model on your dataset

# Build and train an LSTM-based model
max_seq_length = 50
vocab_size = 10000

model_lstm = Sequential()
model_lstm.add(Embedding(input_dim=vocab_size, output_dim=128, input_length=max_seq_length))
model_lstm.add(LSTM(128))
model_lstm.add(Dense(1, activation='sigmoid'))
model_lstm.compile(loss='binary_crossentropy', optimizer='adam')

# Train the LSTM model

# Implement ensemble techniques (e.g., stacking)
def ensemble_predictions(model1, model2, X):
    predictions1 = model1.predict(X)
    predictions2 = model2.predict(X)
    return (predictions1 + predictions2) / 2

# Create an ensemble of your models
ensemble_model = ensemble_predictions(rf_classifier, model_lstm, X_val)

# Evaluate the ensemble model
ensemble_predictions = ensemble_model > 0.5
ensemble_accuracy = accuracy_score(y_val, ensemble_predictions)
print(f"Ensemble Model Accuracy: {ensemble_accuracy}")
```

Diverse Models for Chatbots:

The model can include:

- Traditional Machine Learning Models: Models like decision trees, random forests, or support vector machines are used for prediction.

- Deep Learning Models: Recurrent Neural Networks (RNNs), transformers, or even pre-trained language models like GPT-3 can be part of the ensemble.

Ensemble Strategies:

Several strategies can be employed to combine responses from diverse models. These include:

- Voting: Let each model "vote" on the best response, and select the one with the most votes.
- Weighted Averaging: Assign weights to each model's response based on confidence or accuracy and calculate the final response as a weighted average.
- Stacking: Train another model that takes the responses from individual models as input and produces the chatbot's final response.

Evaluation and Fine-Tuning:

Continuous evaluation of the ensemble chatbot's performance is essential. Metrics like accuracy, coherence, and user satisfaction should guide adjustments to the ensemble strategy and the models used in the ensemble.

DEEP LEARNING ALGORITHM:

Deep learning is a powerful approach for developing chatbots that can understand and generate human-like text responses. Deep learning techniques, especially those involving neural networks, have been widely adopted in chatbot development due to their ability to capture complex patterns in natural language.

It has enabled the development of chatbots that are capable of providing more natural and contextually relevant responses. These chatbots can understand user intent, generate coherent responses, and engage in human-like conversations, making them valuable tools in a wide range of applications, from customer support to virtual assistants.

CONCLUSION:

Thus inorder to innovate a highly enhanced chatbot various machine learning algorithm like ensembling and Deep learning were used. The given dataset is also processed and cleansed according to the needs of training model. The analysis of data is done through different python modules. Therefore, the innovation of chatbot includes training the model with various algorithms and enhancing it further with different combinations of machine learning algorithm.