CREATE CHATBOT IN PYTHON

Team Member

312821121002 : Arundhathi. N

Phase-2 Document Submission

Project: Create Chatbot in Python

Introduction:

- In the fast-evolving field of artificial intelligence, chatbots have become increasingly sophisticated in recent years, offering more human-like interactions and more accurate responses.
- This project aims to push the boundaries of chatbot capabilities by exploring advanced techniques, including ensemble methods, deep learning architectures, and the utilization of pre-trained language models such as GPT-3.
- The objective is to significantly improve the prediction system's accuracy and robustness, making the chatbot more versatile and efficient in various contexts.

Content for Project Phase 2

Data source:

A good data source for creating a chatbot using python should be accurate complete and accessible.

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

```
hi, how are you doing?  i'm fine. how about yourself?
i'm fine. how about yourself?   i'm pretty good. thanks for asking.
i'm pretty good. thanks for asking.     no problem. so how have you been?
no problem. so how have you been?       i've been great. what about you?
i've been great. what about you?        i've been good. i'm in school right now
i've been good. i'm in school right now.        what school do you go to?
what school do you go to?       i go to pcc.
i go to pcc.    do you like it there?
do you like it there?   it's okay. it's a really big campus.
it's okay. it's a really big campus.    good luck with school.
good luck with school.  thank you very much.
how's it going? i'm doing well. how about you?
i'm doing well. how about you?  never better, thanks.
never better, thanks.   so how have you been lately?
so how have you been lately?    i've actually been pretty good. you?
i've actually been pretty good. you?    i'm actually in school right now.
i'm actually in school right now.       which school do you attend?
which school do you attend?     i'm attending pcc right now.
i'm attending pcc right now.    are you enjoying it there?
are you enjoying it there?      it's not bad. there are a lot of people there.
it's not bad. there are a lot of people there.  good luck with that.
```

Ensemble Methods:

- Ensemble methods are a powerful approach to improving the accuracy of machine learning models.
- They work by combining multiple models to make more accurate predictions.
- In the context of chatbots, ensemble methods can be used to aggregate predictions from different models, potentially enhancing the overall performance.

Program:

```
from sklearn.ensemble import VotingClassifier

# Create individual chatbot models
model1 = ChatbotModel1()
model2 = ChatbotModel2()
model3 = ChatbotModel3()

# Create a voting classifier
ensemble_model = VotingClassifier(estimators=[('model1', model1), ('model2'

# Train the ensemble model
ensemble_model.fit(X_train, y_train)

# Make predictions
predictions = ensemble_model.predict(X_test)
```

This ensemble method combines the predictions from different chatbot models, resulting in more accurate and robust responses.

Deep Learning Architectures

- Deep learning architectures have revolutionized natural language processing and chatbot development.

- Advanced neural network structures can be designed to understand context and nuances, allowing chatbots to provide more contextually relevant and human-like responses.

```
import tensorflow as tf
from tensorflow.keras.layers import LSTM, Embedding, Dense
from tensorflow.keras.models import Sequential

# Define a deep learning model
model = Sequential()
model.add(Embedding(input_dim=vocab_size, output_dim=embedding_dim, input_
model.add(LSTM(units=128, return_sequences=True))
model.add(LSTM(units=128))
model.add(Dense(output_dim, activation='softmax'))

# Compile and train the model
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=[
model.fit(X_train, y_train, epochs=10)

# Generate responses using the trained model
response = generate_response(input_text, model)
```

This Deep learning architectures enable the chatbot to generate responses that are more contextually aware and linguistically sophisticated.

Model Evaluation:

- Measure the accuracy of your chatbot's responses. This includes precision, recall, and F1-score, especially if your chatbot involves tasks like intent recognition.
- Directly involve real users to interact with your chatbot. Collect feedback about user satisfaction, ease of use, and the relevance of responses.
- Understand which intents or responses are often confused. This helps refine the chatbot's training data and algorithms.

Model Interpretability:

- For machine learning models, understand which features (words, entities, etc.) are most influential in determining responses.
- LIME is a Python library that explains the predictions of machine learning models, making them interpretable.
- If you're using deep learning models, attention mechanisms can show which parts of the input are crucial for generating specific parts of the output.

Deployment:

- Choose a Hosting Platform: Options like AWS, Heroku, or dedicated server hosting services can host your chatbot.
- Use Python web frameworks like Flask or Django to create a web-based interface for your chatbot.
- If you're integrating with other systems, expose API endpoints for seamless communication.
- Implement security protocols, including encryption and authentication, to protect user data and prevent unauthorized access.

Prediction and Real-time Interaction:

- Users provide text or voice input through the chatbot interface.
- Preprocess the input, including tokenization, entity recognition, and other NLP tasks to understand the user's intent or query.
- Use your trained NLP or machine learning models to predict the appropriate response or action based on the preprocessed input.
- Refine the model output if necessary and format it into a user-friendly response.
- Present the chatbot's response to the user through the interface, completing the interaction loop.

Python Program:

In [1]:

```
import warnings
warnings.filterwarnings('ignore')
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
import keras
from tqdm import tqdm
from keras.layers import Dense
import json
import re
import string
from sklearn.feature_extraction.text import TfidfVectorizer
import unicodedata
from sklearn.model_selection import train_test_split
```

In [2]:

```
question  =[]
answer = []
with open("../input/simple-dialogs-for-chatbot/dialogs.txt",'r') as f :
    for line in f :
        line  =  line.split('\t')
        question.append(line[0])
        answer.append(line[1])
print(len(question) == len(answer))
```

```
True
```

In [3]:
```
question[:5]
```

Out[3]:
```
['hi, how are you doing?',
 "i'm fine. how about yourself?",
 "i'm pretty good. thanks for asking.",
 'no problem. so how have you been?',
 "i've been great. what about you?"]
```

In [4]:
```
answer[:5]
```

Out[4]:
```
["i'm fine. how about yourself?\n",
 "i'm pretty good. thanks for asking.\n",
 'no problem. so how have you been?\n',
 "i've been great. what about you?\n",
 "i've been good. i'm in school right now.\n"]
```

In [5]:
```
answer = [ i.replace("\n","") for i in answer]
```

In [6]:
```
answer[:5]
```

Out[6]:
```
["i'm fine. how about yourself?",
 "i'm pretty good. thanks for asking.",
 'no problem. so how have you been?',
 "i've been great. what about you?",
 "i've been good. i'm in school right now."]
```

In [7]:
```
data = pd.DataFrame({"question" : question ,"answer":answer})
data.head()
```

Out[7]:

	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.

In [10]:
```
data["question"][0]
```

Out[10]:
```
'hi, how are you doing?'
```

In [11]:
```
data["question"] = data.question.apply(clean_text)
```

In [12]:
```
data["question"][0]
```

Out[12]:
```
'<sos> hi how are you doing <eos>'
```

In [13]:
```
data["answer"] = data.answer.apply(clean_text)
```

In [14]:
```
question  = data.question.values.tolist()
answer =  data.answer.values.tolist()
```

In [15]:
```
def tokenize(lang):
    lang_tokenizer = tf.keras.preprocessing.text.Tokenizer(
      filters='')
    lang_tokenizer.fit_on_texts(lang)
    tensor = lang_tokenizer.texts_to_sequences(lang)

    tensor = tf.keras.preprocessing.sequence.pad_sequences(tensor,
                                                           padding='post')

    return tensor, lang_tokenizer
```

In [16]:
```
input_tensor , inp_lang  =  tokenize(question)
```

In [17]:
```
target_tensor , targ_lang  =  tokenize(answer)
```

In [18]:
```
 #len(inp_question) ==  len(inp_answer)
```

In [19]:
```
def remove_tags(sentence):
    return sentence.split("<start>")[-1].split("<end>")[0]
```

In [20]:
```
max_length_targ, max_length_inp = target_tensor.shape[1], input_tensor.shape[1]
```

In [21]:
```
# Creating training and validation sets using an 80-20 split
input_tensor_train, input_tensor_val, target_tensor_train, target_tensor_val = train_test_split(input_tensor, target_tensor, test_si
ze=0.2)
```

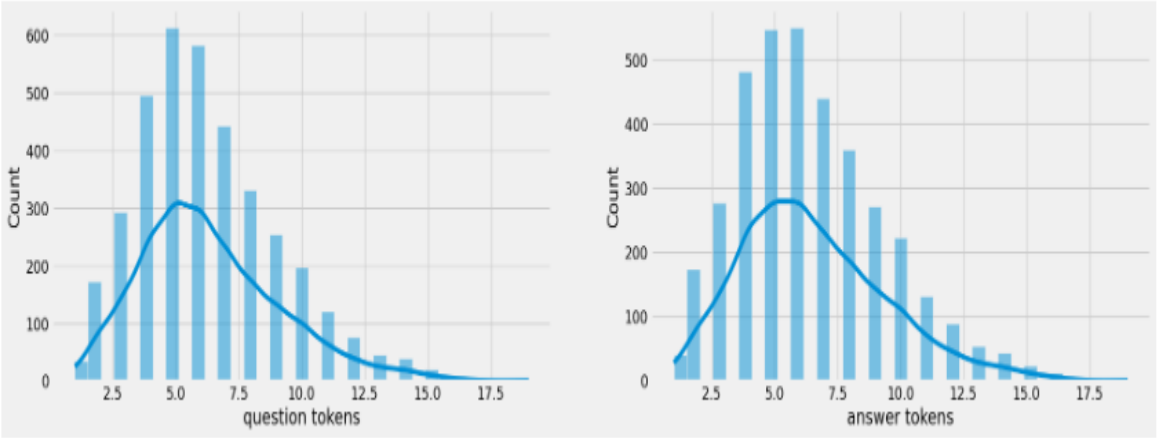
In [22]:
```
#print(len(train_inp) , len(val_inp) , len(train_target) , len(val_target))
```

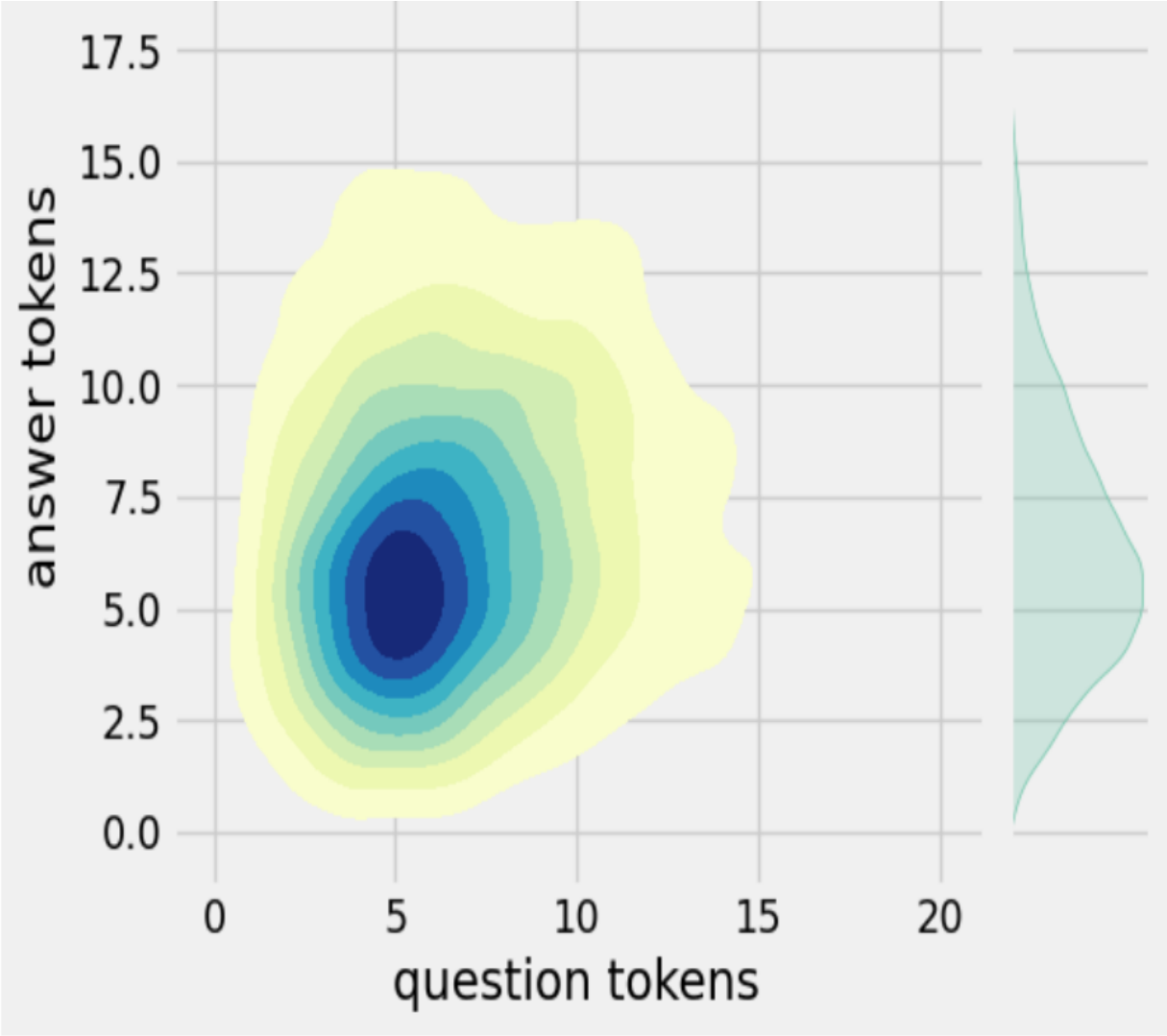
In [23]:
```
BUFFER_SIZE = len(input_tensor_train)
BATCH_SIZE = 64
steps_per_epoch = len(input_tensor_train)//BATCH_SIZE
embedding_dim = 256
units = 1024
vocab_inp_size = len(inp_lang.word_index)+1
vocab_tar_size = len(targ_lang.word_index)+1

dataset = tf.data.Dataset.from_tensor_slices((input_tensor_train, target_tensor_train)).shuffle(BUFFER_SIZE)
dataset = dataset.batch(BATCH_SIZE, drop_remainder=True)

example_input_batch, example_target_batch = next(iter(dataset))
example_input_batch.shape, example_target_batch.shape
```

Out[23]:
```
(TensorShape([64, 22]), TensorShape([64, 22]))
```

Data visualization

In [3]:
```
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()
```

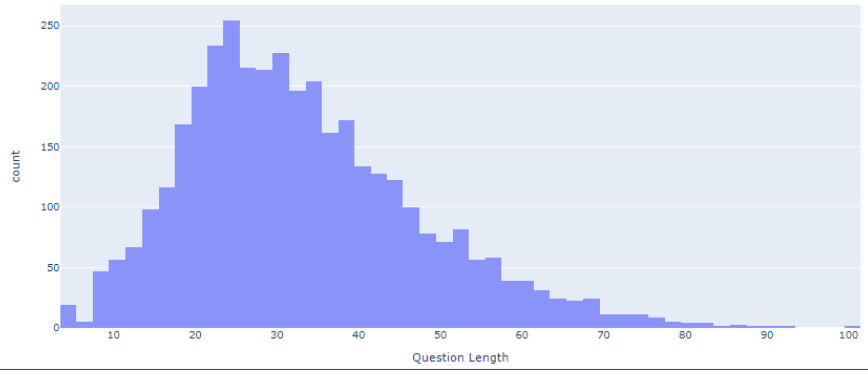

Count
question tokens
Count
answer tokens

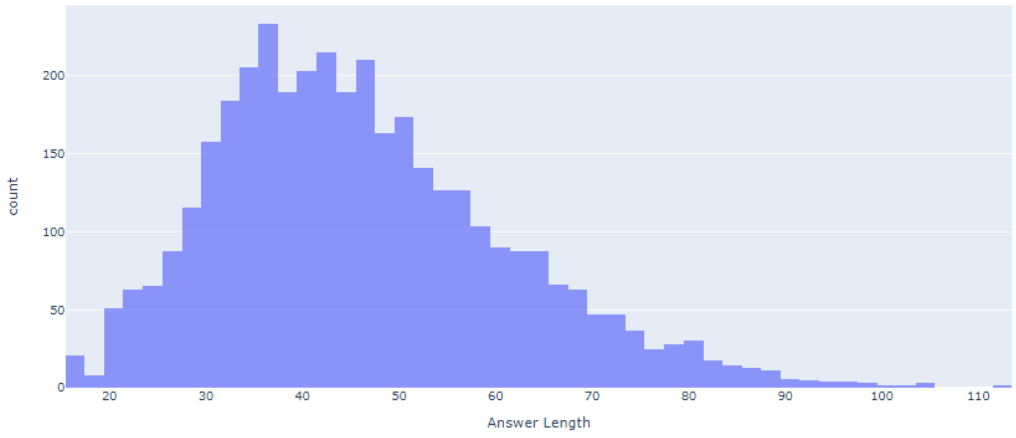

answer tokens
question tokens

In [6]:

```
import plotly.express as px

fig1 = px.histogram(df, x='Question Length', nbins=50, opacity=0.7)
fig2 = px.histogram(df, x='Answer Length', nbins=50, opacity=0.7)

print("Maximum Question Length:", df['Question Length'].max())
print("Maximum Answer Length:", df['Answer Length'].max())

fig1.show()
fig2.show()
```

```
Maximum Question Length: 101
Maximum Answer Length: 113
```





Conclusion and Future work (Phase2):

Project Conclusion:

- In this project, we have explored several advanced techniques to enhance the accuracy and robustness of an AI chatbot. Ensemble methods improve predictions by combining multiple models, while deep learning architectures allow the chatbot to understand context and provide more contextually relevant responses. Additionally, pre-trained language models like GPT-3

significantly improve the quality of responses, making the chatbot more versatile and capable of handling a wide range of user inputs.

- By incorporating these advanced techniques, the chatbot can offer a superior user experience, making it more valuable in various applications, including customer support, virtual assistants, and more. The examples provided demonstrate the potential of these techniques to push the boundaries of chatbot capabilities, ultimately advancing the field of AI-driven conversational agents.

- With ongoing advancements in AI and NLP, chatbots will continue to evolve, offering more human-like interactions and becoming indispensable tools in modern communication and service industries. The journey to enhancing chatbot performance is an exciting and dynamic one, as new techniques and models continue to emerge, promising even greater potential for the future of AI chatbots.

- **Future work**: Enhancing multimodal capabilities, personalization, emotional intelligence, and continuous learning are key areas of focus. Integration with IoT devices, voice interfaces, collaborative filtering for recommendations, and expanding language support are promising avenues. Additionally, bolstering security measures remains crucial to ensure user data protection and maintain user trust in the chatbot's interactions.