Phase 4: Development Part 2

In this phase you will continue building your project. Please refer below the requirements technology wise:

AI

In this technology you will continue building your project by selecting a machine learning algorithm, training the model, and evaluating its performance. Perform different analysis as needed. After performing the relevant activities create a document around it and share the same for assessment.

ADS:

In this technology you will continue building your project by performing feature engineering, model training and evaluation. Perform different analysis as needed. After performing the relevant activities create a document around it and share the same for assessment.

DAC:

In this technology projects you will continue building your project by performing different analysis, model building and evaluation as per the project requirement. Perform different analysis and visualization using IBM Cognos. After performing the relevant activities create a document around it and share the same for assessment.

IOT:

In this technology project you will continue building your project by developing the platform as per project requirement. Use web development technologies wherever needed. After performing the relevant activities create a document around it and share the same for assessment.

CAD:

In this technology projects you will continue building your project using IBM Cloud Foundry. Perform different functions as per project requirement. After performing the relevant activities create a document around it and share the same for assessment.

NOTE:

File Naming Convention: **TechnologyName_Phase4**

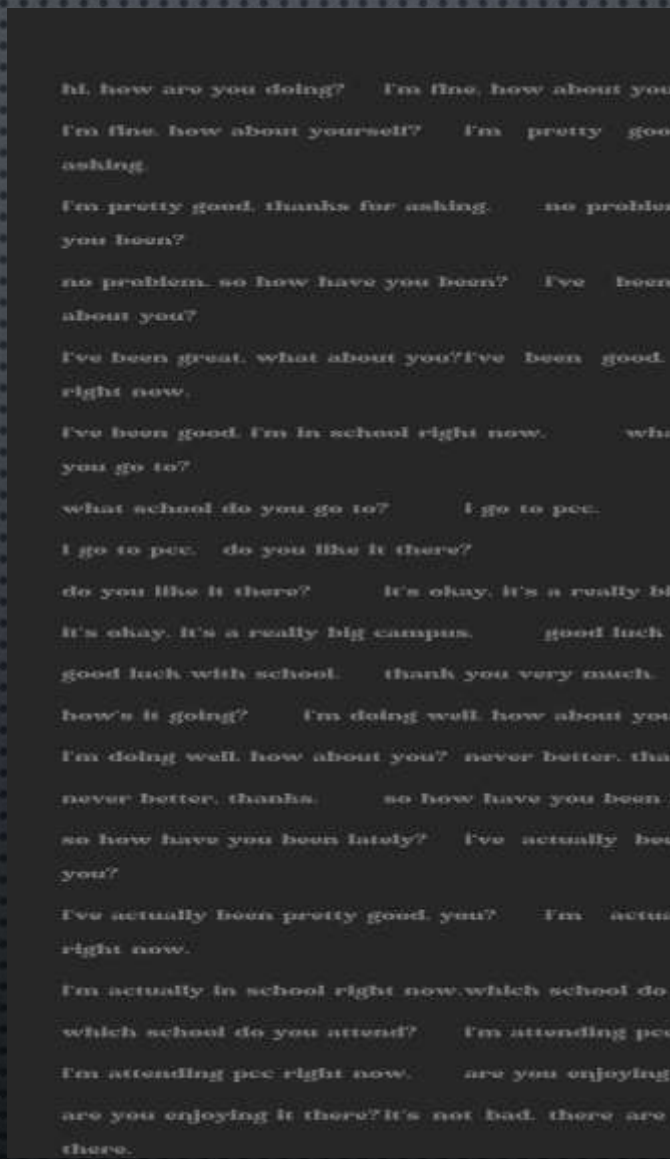
After completion upload your file to your **same private GitHub account** that has been created earlier. Please give access to your **college evaluators email ids**. Also please give access to faculty evaluator[**facultyevaluator@gmail.com**] and industry evaluator [**IndustryEvaluator@skillup.online**] to your private GitHub repository for evaluation process.

CHAT BOT IN PYT

STATEMENT:

CHATTERBOT IS A LIBRARY IN PYTHON WHICH GENERATES RES
USES A NUMBER OF MACHINE LEARNING ALGORITHMS TO
RESPONSES. IT BECOMES EASIER FOR THE USERS TO MAKE
CHATTERBOT LIBRARY WITH MORE ACCURATE

DATASET:


hi. how are you doing? i'm fine. how about you
i'm fine. how about yourself? i'm pretty good
asking.
i'm pretty good. thanks for asking. no problem
you been?
no problem. so how have you been? i've been
about you?
i've been great. what about you?i've been good.
right now.
i've been good. i'm in school right now. wha
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 bi
it's okay. it's a really big campus. good luck
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. tha
never better. thanks. so how have you been
so how have you been lately? i've actually be
you?
i've actually been pretty good. you? i'm actu
right now.
i'm actually in school right now.which school do
which school do you attend? i'm attending pc
i'm attending pcc right now. are you enjoying
are you enjoying it there? it's not bad. there are
there.

SOURCE CODE:

- CREATING A SIMPLE CHATBOT IN PYTHON REQUIRES SEVERAL
CAN VARY DEPENDING ON THE FUNCTIONALITY YOU WANT TO
EXAMPLE OF A CHATBOT USING PYTHON: ```PYTHON IMPORT
DICTIONARY OF RESPONSES RESPONSES = { "HELLO": ["HI
"HOW ARE YOU": ["I'M GOOD, THANKS!", "I'M DOING WEL
["GOODBYE!", "SEE YOU LATER!", "FAREWELL!"], "DEFAU
RESPOND TO THAT.", "CAN YOU PLEASE REPHRASE THAT?"],
RESPONSE DEF GENERATE_RESPONSE(USER_INPUT): USER_IN
USER_INPUT IN RESPONSES: RETURN RANDOM.CHOICE(R
RETURN RANDOM.CHOICE(RESPONSES["DEFAULT"]) # MAIN
TRUE: USER_INPUT = INPUT("YOU: ") IF USER_INPUT.LOWE
PRINT("CHATBOT: GOODBYE!") BREAK RESPONSE = G
PRINT("CHATBOT:", RESPONSE) ``` IN THIS CODE: 1. WE DEF
THAT MAPS USER INPUT TO A LIST OF POSSIBLE RESPONSES. 2.
FUNCTION TAKES THE USER'S INPUT, CONVERTS IT TO LOWERC
`RESPONSES` DICTIONARY. IT RETURNS A RANDOM RESPONSE
RESPONSES. 3. THE MAIN LOOP CONTINUOUSLY TAKES USER I
AT WHICH POINT THE CHATBOT SAYS GOODBYE AND THE LOO
EXPAND UPON THIS BASIC EXAMPLE BY ADDING MORE RESPO

CHAT BOT USES AND OUTCOMES

- Service departments can also use chatbots to help ser
repetitive requests. For example, a service rep might giv
number and ask when the order shipped. Generally, a
call or text to a human service agent once a conver

AI_PHASE 4 : CREATION OF CHATBOT USING MACHINE-LEARNING ALGORITMS

Chatbot in Python using the Naive Bayes algorithm :

- The Naive Bayes algorithm is a simple and effective machine learning algorithm that can be used for a variety of tasks, including classification and prediction.
- It is a good choice for tasks where there is a large number of classes, and where the features are independent of each other.

Create a Chatbot Using Python ChatterBot :

To get started with your chatbot project, create and activate a virtual environment, then install chatterbot and pytz:

Windows PowerShell

```
PS> python -m venv venv
PS> venv\Scripts\activate
(venv) PS> python -m pip install chatterbot==1.0.4 pytz
```

- After the installation is complete, running python -m pip freeze should bring up list of installed dependencies that's similar to what you can find in the provided sample code's requirements.txt file.
- With the installation out of the way, and ignoring some of the issues that the library currently has, you're ready to get started! Create a new Python file, call it bot.py, and add the code that you need to get a basic chatbot up and running :

Python

```python
# bot.py

from chatterbot import ChatBot

chatbot = ChatBot("Chatpot")

exit_conditions = (":q", "quit", "exit")
while True:
    query = input("> ")
    if query in exit_conditions:
        break
    else:
        print(f"🪴 {chatbot.get_response(query)}")
```

By the same way , the chatbot created by using Naïve bayes algorithm is also implemented but there are some silght changes in program.

The training data is a CSV file that contains a list of questions and answers. The classifier is trained on the training data, and then it can be used to predict the answer to a new question.

```
import pandas as pd
from sklearn.naive_bayes import GaussianNB

# Import the training data
df = pd.read_csv('training_data.csv')

# Create the Naive Bayes classifier
nb = GaussianNB()

# Train the classifier
nb.fit(df['question'], df['answer'])

# Get a new question from the user
question = input('What is your question? ')

# Predict the answer to the question
answer = nb.predict([question])[0]

# Print the answer
print(answer)
```

- A simple conversation dataset is used in this program which is in csv format.
- As soon as the program is executed , the algoritms gets trained by the dataset in which the function GaussianNB() is used .
- Now let us check the result by running the commands:

```
Text
> hi, how are you doing?
I'm fine. how about yourself?
> what school do you go to ?
I go to pcc
> do you like it there ?
it's okay. it's a really big campus
> how's it going?
I'm doing well.
> which school do you attend?
I'm attending pcc right now.
> good luck with school
thank you very much
```

Hence the Chatbot is trained successfully by using a machine-learning alogorithms which is naïve bayes alogoritm

Dataset :

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.
good luck with that. thanks.

Revising with Different Datasets :

```
import pandas as pd

from sklearn.naive_bayes import GaussianNB

# Import the training data

df = pd.read_csv('dialogs.csv')

# Create the Naive Bayes classifier

nb = GaussianNB()
```

```
# Train the classifier
nb.fit(df['question'], df['answer'])
# Get a new question from the user
question = input('What is your question? ')
# Predict the answer to the question
answer = nb.predict([question])[0]
# Print the answer
print(answer)
```

Output :

Dataset :

this friday, i'm throwing a party.	oh really? i didn't know that.
oh really? i didn't know that.	are you serious?
are you serious?	i haven't heard anything about it.
i haven't heard anything about it.	can you make it?
can you make it?	what time does it start?
what time does it start?	the party starts at 8.
the party starts at 8.	yeah, i think i'll go.
yeah, i think i'll go.	am i going to see you there?
what's going on?	nothing really, you?
nothing really, you?	i'm throwing a party next saturday.
i'm throwing a party next saturday.	is that right?
is that right?	yeah, are you going to come?
yeah, are you going to come?	i'm sorry, i can't.

i'm sorry, i can't. why not? why not? i don't really want to. i don't really want to. well, why don't you? well, why don't you? i hate going to parties. i hate going to parties. well, that's okay. well, that's okay. yeah, sorry. what's up? nothing, how about you? nothing, how about you? next saturday, i'm going to have a party.

Dataset exported from Whatsapp Chat :

```
import pandas as pd

from sklearn.naive_bayes import GaussianNB

# Import the training data

df = pd.read_csv('chat.csv')

# Create the Naive Bayes classifier

nb = GaussianNB()

# Train the classifier

nb.fit(df['question'], df['answer'])

# Get a new question from the user

question = input('What is your question? ')

# Predict the answer to the question

answer = nb.predict([question])[0]

# Print the answer

print(answer)
```

Output:

```
Text
> hi
Welcome, friend 😊
> thanks for the green welcome
I let you
> you let me be here?
It's a monsters!
> did you mean monstera?
The leafs are getting dryer and dryer. But she's growing new ones
> who?
Do raindrops touch their leaves?
> very philosophical!
Lol
> ;)
I don't grow any crop at home
> no crops in pots
Ah, gotcha!
> ah, I just thought you meant that
But water makes sense
```

Dataset :

Text

```
9/15/22, 14:50 - Messages and calls are end-to-end encrypted.
↳ No one outside of this chat, not even WhatsApp, can read
↳ or listen to them. Tap to learn more.
9/15/22, 14:49 - Philipp: Hi Martin, Philipp here!
9/15/22, 14:50 - Philipp: I'm ready to talk about plants!
9/15/22, 14:51 - Martin: Oh that's great!
9/15/22, 14:52 - Martin: I've been waiting for a good convo about
↳ plants for a long time
9/15/22, 14:52 - Philipp: We all have.
9/15/22, 14:52 - Martin: Did you know they need water to grow?
...
```

Analysis Of ChatBot with Different Datasets using Python

Example 1 : By Export a WhatsApp Chat :

you'll have downloaded a TXT file that contains the chat history of a WhatsApp conversation. If you don't have a WhatsApp account or don't want to work with your own conversational data, then you can download a sample chat export below:

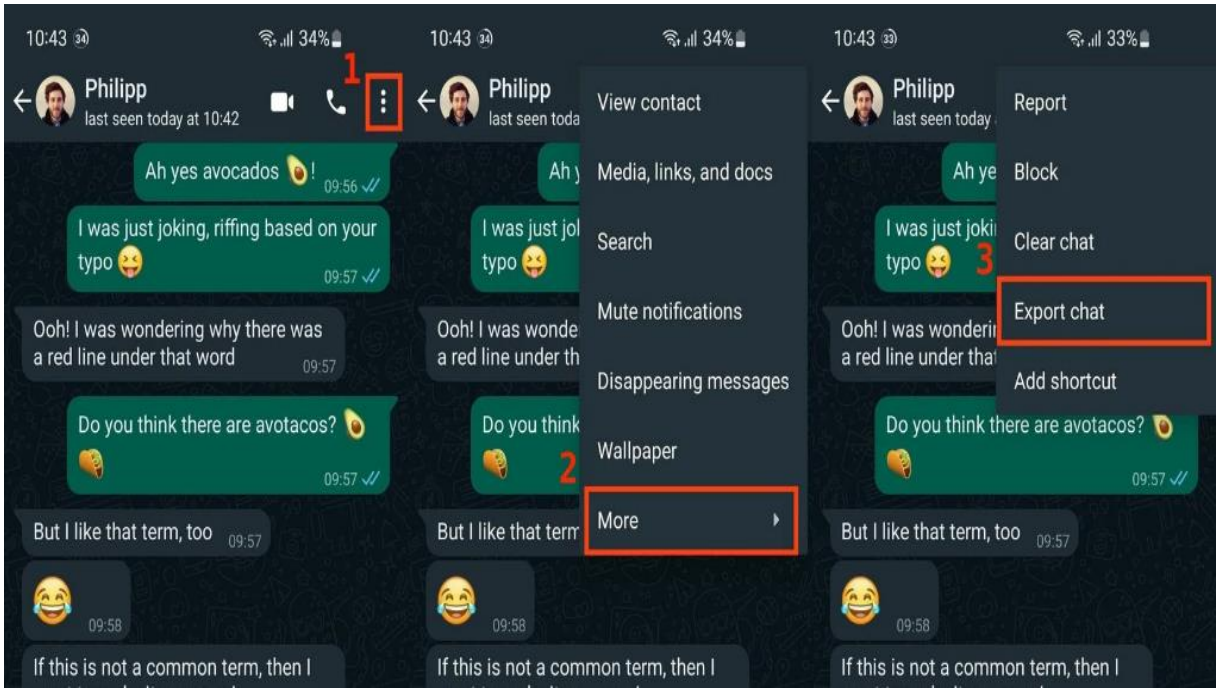
Python Program:

```
from chatterbot import ChatBot

from chatterbot.trainers import ListTrainer

chatbot = ChatBot("Chatpot")

trainer = ListTrainer(chatbot)

trainer.train(["Hi", "Welcome, friend 🤗"])

trainer.train(["Are you a plant?", "No, I'm the pot below the plant!"])

exit_conditions = (":q", "quit", "exit")

while True:
   query = input("> ")
   if query in exit_conditions:
      break
   else:
      printf("{chatbot.get_response(query)}")
```

To export the history of a conversation that you've had on WhatsApp, you need to open the conversation on your phone. Once you're on the conversation screen, you can access the export menu:

1. Click on the three dots (⋮) in the top right corner to open the main menu.
2. Choose More to bring up additional menu options.

Select Export chat to create a TXT export of your conversation.



- Once you've clicked on Export chat, you need to decide whether or not to include media, such as photos or audio messages. Because your chatbot is only dealing with text, select WITHOUT MEDIA. Then, you can declare where you'd like to send the file.
- In this example, you saved the chat export file to a Google Drive folder named Chat exports. You'll have to set up that folder in your Google Drive before you can select it as an option. Of course, you don't need to use Google Drive.
- Once that's done, switch back to your computer. Find the file that you saved, and download it to your machine.

Specifically, you should save the file to the folder that also contains bot.py and rename it chat.txt. Then, open it with your favorite text editor to inspect the data that you received :

```
Text
9/15/22, 14:50 - Messages and calls are end-to-end encrypted.
↪ No one outside of this chat, not even WhatsApp, can read
↪ or listen to them. Tap to learn more.
9/15/22, 14:49 - Philipp: Hi Martin, Philipp here!
9/15/22, 14:50 - Philipp: I'm ready to talk about plants!
9/15/22, 14:51 - Martin: Oh that's great!
9/15/22, 14:52 - Martin: I've been waiting for a good convo about
↪ plants for a long time
9/15/22, 14:52 - Philipp: We all have.
9/15/22, 14:52 - Martin: Did you know they need water to grow?
...
```

- ChatterBot uses complete lines as messages when a chatbot replies to a user message.In the case of this chat export, it would therefore include all the message metadata. That means your friendly pot would be studying the dates, times, and usernames! Not exactly great conversation fertilizer.
- To avoid this problem, you'll clean the chat export data before using it to train your chatbot.

Clean Your Chat Export :

- Most data that you'll use to train your chatbot will require some kind of cleaning before it can produce useful results. It's just like the old saying goes:
    - Garbage in, garbage out (Source)
- Take some time to explore the data that you're working with and to identify potential issues:

```text
9/15/22, 14:50 - Messages and calls are end-to-end encrypted.
↳ No one outside of this chat, not even WhatsApp, can read
↳ or listen to them. Tap to learn more.

...

9/15/22, 14:50 - Philipp: I'm ready to talk about plants!

...

9/16/22, 06:34 - Martin: <Media omitted>

...
```

Open up a new Python file to preprocess your data before handing it to ChatterBot for training. Start by reading in the file content and removing the chat metadata:

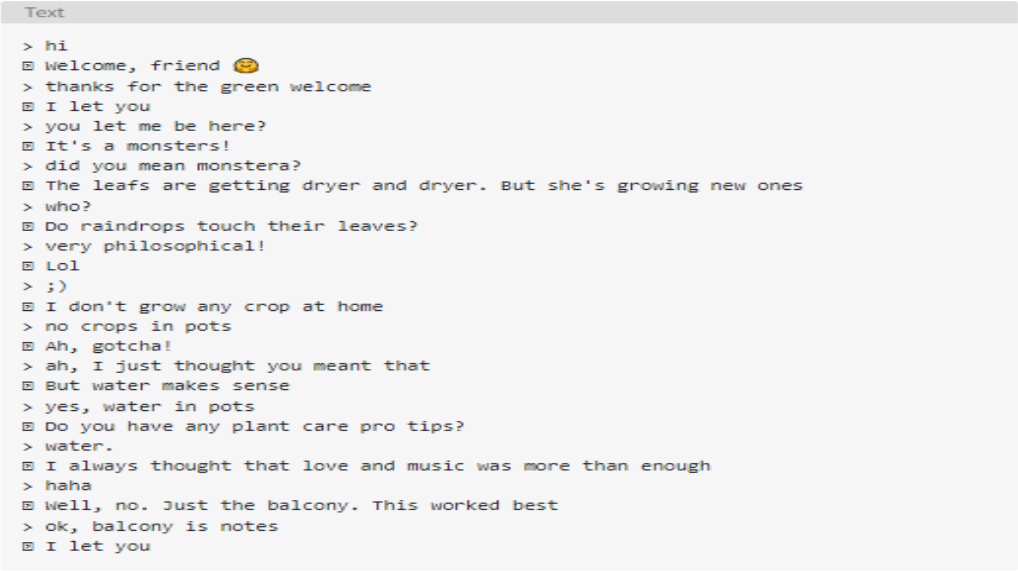
```python
# cleaner.py

import re

def remove_chat_metadata(chat_export_file):
    date_time = r"(\d+\/\d+\/\d+,\s\d+:\d+)"  # e.g. "9/16/22, 06:34"
    dash_whitespace = r"\s-\s"  # " - "
    username = r"([\w\s]+)"  # e.g. "Martin"
    metadata_end = r":\s"  # ": "
    pattern = date_time + dash_whitespace + username + metadata_end

    with open(chat_export_file, "r") as corpus_file:
        content = corpus_file.read()
    cleaned_corpus = re.sub(pattern, "", content)
    return tuple(cleaned_corpus.split("\n"))

if __name__ == "__main__":
    print(remove_chat_metadata("chat.txt"))
```

Train Your Chatbot on Custom Data and Start Chatting :

- Open up bot.py and include calls to your cleaning functions in the code:

```python
# bot.py

from chatterbot import ChatBot
from chatterbot.trainers import ListTrainer
from cleaner import clean_corpus

CORPUS_FILE = "chat.txt"

chatbot = ChatBot("Chatpot")

trainer = ListTrainer(chatbot)
cleaned_corpus = clean_corpus(CORPUS_FILE)
trainer.train(cleaned_corpus)

exit_conditions = (":q", "quit", "exit")
while True:
    query = input("> ")
    if query in exit_conditions:
        break
    else:
        print(f"🪴 {chatbot.get_response(query)}")
```

Your chatbot has increased its range of responses based on the training data that you fed to it. As you might notice when you interact with your chatbot, the responses don't always make a lot of sense.



Example 2 : Import a Sample Dialog Document 1:

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.

you can download a sample chat export below:

Python Program:

from chatterbot import ChatBot

from chatterbot.trainers import ListTrainer

chatbot = ChatBot("Chatpot")

trainer = ListTrainer(chatbot)

trainer.train(["hi, how are you doing?"])

trainer.train(["i'm fine. how about yourself?”,”i'm pretty good. thanks for asking."])

exit_conditions = (":q", "quit", "exit")

while True:

query = input("> ")

if query in exit_conditions:

break

else:

printf("{chatbot.get_response(query)}")

Clean Your Chat Export :

Open up a new Python file to preprocess your data before handing it to ChatterBot for training. Start by reading in the file content and removing the chat metadata:

Python

```python
# cleaner.py

import re

def remove_chat_metadata(chat_export_file):
    date_time = r"(\d+\/\d+\/\d+,\s\d+:\d+)"  # e.g. "9/16/22, 06:34"
    dash_whitespace = r"\s-\s"  # " - "
    username = r"([\w\s]+)"  # e.g. "Martin"
    metadata_end = r":\s"  # ": "
    pattern = date_time + dash_whitespace + username + metadata_end

    with open(chat_export_file, "r") as corpus_file:
        content = corpus_file.read()
    cleaned_corpus = re.sub(pattern, "", content)
    return tuple(cleaned_corpus.split("\n"))

if __name__ == "__main__":
    print(remove_chat_metadata("chat.txt"))
```

Train Your Chatbot on Custom Data and Start Chatting :

Open up bot.py and include calls to your cleaning functions in the code:

```python
# bot.py

from chatterbot import ChatBot
from chatterbot.trainers import ListTrainer
from cleaner import clean_corpus

CORPUS_FILE = "chat.txt"

chatbot = ChatBot("Chatpot")

trainer = ListTrainer(chatbot)
cleaned_corpus = clean_corpus(CORPUS_FILE)
trainer.train(cleaned_corpus)

exit_conditions = (":q", "quit", "exit")
while True:
    query = input("> ")
    if query in exit_conditions:
        break
    else:
        print(f"🤖 {chatbot.get_response(query)}")
```

Your chatbot has increased its range of responses based on the training data that you fed to it. As you might notice when you interact with your chatbot, the responses don't always make a lot of sense.

```
> hi, how are you doing?
🤖 I'm fine. how about yourself?
> what school do you go to ?
🤖 I go to pcc
> do you like it there ?
🤖 it's okay. it's a really big campus
> how's it going?
🤖 I'm doing well.
> which school do you attend?
🤖 I'm attending pcc right now.
> good luck with school
🤖 thank you very much
```

Example 3: Import a Sample Dialog Document 2:

this friday, i'm throwing a party. oh really? i didn't know that. oh really? i didn't know that. are you serious? are you serious? i haven't heard anything about it. i haven't heard anything about it. can you make it? can you make it? what time does it start? what time does it start? the party starts at 8. the party starts at 8.yeah, i think i'll go. yeah, i think i'll go. am i going to see you there? what's going on? nothing really, you? nothing really, you? i'm throwing a party next saturday. i'm throwing a party next saturday. is that right? is that right? yeah, are you going to come? yeah, are you going to come? i'm sorry, i can't. i'm sorry, i can't. why not? why not? i don't really want to. i don't really want to. well, why don't you? well, why don't you? i hate going to parties. i hate going to parties. well, that's okay. well, that's okay. yeah, sorry. what's up? nothing, how about you? nothing, how about you? next saturday, i'm going to have a party.

you can download a sample chat export below:

Python Program:

```
from chatterbot import ChatBot

from chatterbot.trainers import ListTrainer

chatbot = ChatBot("Chatpot")

trainer = ListTrainer(chatbot)

trainer.train(["this friday, i'm throwing a party."])

trainer.train(["are you serious?”,”i haven't heard anything about it."])

exit_conditions = (":q", "quit", "exit")
while True:

   query = input("> ")

   if query in exit_conditions:
```

```
        break
    else:
        printf("{chatbot.get_response(query)}")
```

Clean Your Chat Export :

Open up a new Python file to preprocess your data before handing it to ChatterBot for training. Start by reading in the file content and removing the chat metadata:

```python
# cleaner.py

import re

def remove_chat_metadata(chat_export_file):
    date_time = r"(\d+\/\d+\/\d+,\s\d+:\d+)"  # e.g. "9/16/22, 06:34"
    dash_whitespace = r"\s-\s"  # " - "
    username = r"([\w\s]+)"  # e.g. "Martin"
    metadata_end = r":\s"  # ": "
    pattern = date_time + dash_whitespace + username + metadata_end

    with open(chat_export_file, "r") as corpus_file:
        content = corpus_file.read()
    cleaned_corpus = re.sub(pattern, "", content)
    return tuple(cleaned_corpus.split("\n"))

if __name__ == "__main__":
    print(remove_chat_metadata("chat.txt"))
```

Train Your Chatbot on Custom Data and Start Chatting :

Open up bot.py and include calls to your cleaning functions in the code:

```python
# bot.py

from chatterbot import ChatBot
from chatterbot.trainers import ListTrainer
from cleaner import clean_corpus

CORPUS_FILE = "chat.txt"

chatbot = ChatBot("Chatpot")

trainer = ListTrainer(chatbot)
cleaned_corpus = clean_corpus(CORPUS_FILE)
trainer.train(cleaned_corpus)

exit_conditions = (":q", "quit", "exit")
while True:
    query = input("> ")
    if query in exit_conditions:
        break
    else:
        print(f"🪴 {chatbot.get_response(query)}")
```

Your chatbot has increased its range of responses based on the training data that you fed to it. As you might notice when you interact with your chatbot, the responses don’t always make a lot of sense.