CREATE A CHATBOT IN PYTHON

TEAM MEMBER

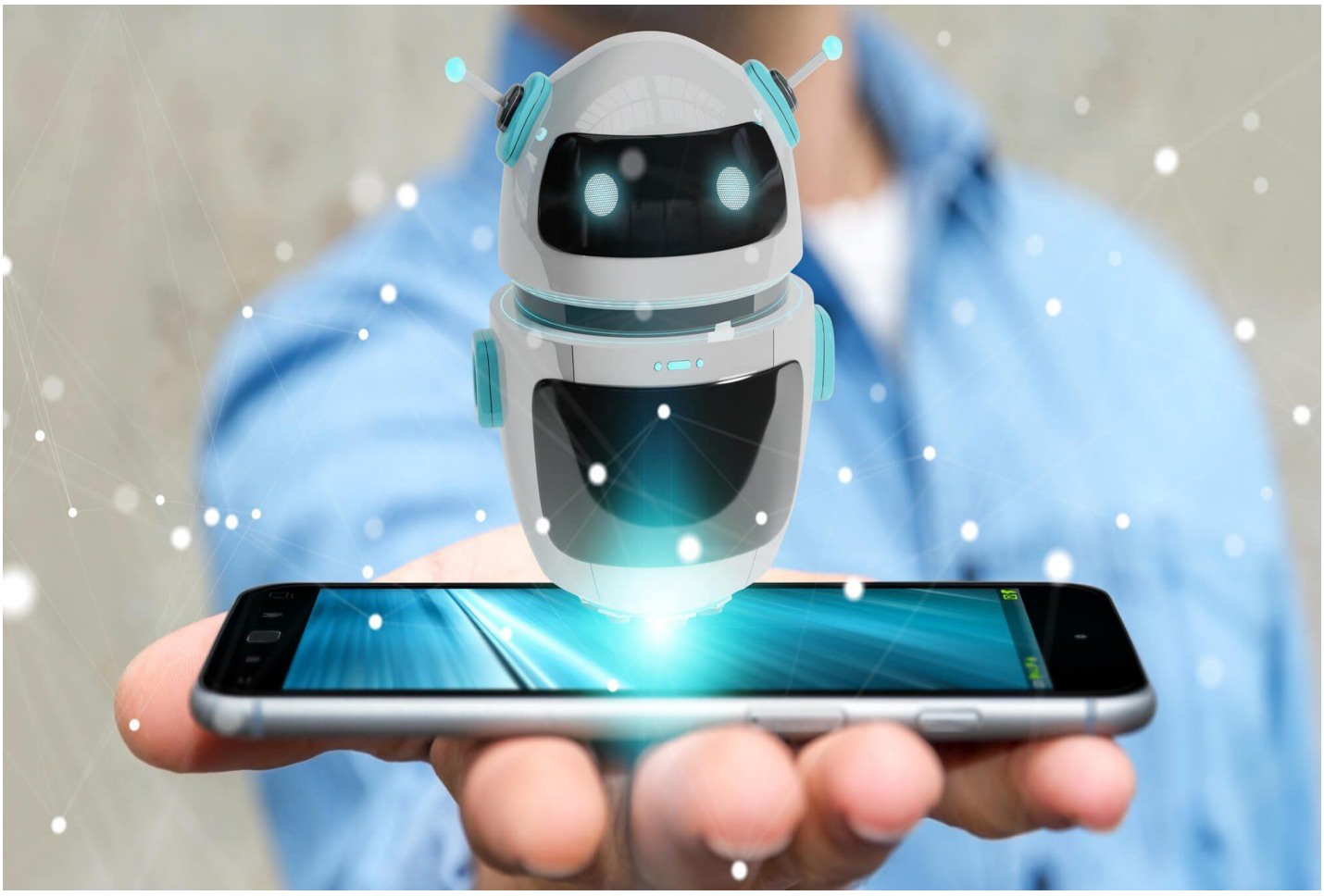
810621104009: JAYAKODI.D

PHASE-3 SUBMISSION DOCUMENT

PROJECT TITLE: CREATE A CHATBOT IN PYTHON

PHASE 3: Development part 1

Topic: Start building the create a chatbot in python model by using loading and pre-processing the dataset.



INTRODUCTION :

In recent years, the adoption and use cases of chatbots have been on the rise. With advancements in Natural Language Processing (NLP) and the introduction of models like ChatGPT, chatbots have become increasingly popular and powerful tools for automating conversations.

In this article, we will explore the process of creating a simple chatbot using Python and NLP techniques. Whether you're interested in building a virtual assistant, customer support bot, or simply want to explore the fascinating world of chatbots, this article will guide you through the steps of creating your own. So, let's dive in and harness the potential of chatbots in this era of technological advancement.

Preprocessing User Input :

Before we can process user input, we need to preprocess it by:

- **Tokenization**: the process of breaking down a sentence or text into individual words or tokens. In this step, we use the `nltk.word_tokenize()` function from the NLTK library to split the user input into a list of tokens. For example, the sentence *"How are*

you doing today?" would be tokenized into `['How', 'are', 'you', 'doing', 'today', '?']`.

1. **Lowercasing:** to ensure consistency and remove case sensitivity, we convert all tokens to lowercase using the `lower()` method. This helps in matching words regardless of whether they are in uppercase or lowercase.
2. **Lemmatization**: it reduces words to their base or dictionary form, known as the lemma. It helps in reducing inflected forms to their base form and normalizes the words. We use the `WordNetLemmatizer` class from the NLTK library to perform lemmatization on each token. For example, the word "*running*" would be lemmatized to "*run*," and "*better*" would remain as it is.
3. **Joining Tokens**: the last step is to join the lemmatized tokens back into a single string. This step is necessary as many NLP techniques and algorithms expect input in the form of a string rather than a list of tokens. We use the `' '.join()` method to concatenate the tokens with a space in between, resulting in a preprocessed user input string.

```
def preprocess_input(user_input):
    lemmatizer = WordNetLemmatizer()
    tokens = nltk.word_tokenize(user_input.lower())
    lemmatized_tokens = [lemmatizer.lemmatize(token) for token in tokens]
    return ' '.join(lemmatized_tokens)
```

Building the Chatbot Core

The chatbot core includes creating intent recognition, entity extraction, and response generation components. In this example, we will focus on a simple response generation mechanism using the TF-IDF vectorization technique and cosine similarity for matching user input with predefined responses.

```
def generate_response(user_input, corpus):
    tfidf_vectorizer = TfidfVectorizer()
    tfidf_matrix = tfidf_vectorizer.fit_transform(corpus)
    user_input = preprocess_input(user_input)
    user_input_vector = tfidf_vectorizer.transform([user_input])
    similarities = cosine_similarity(user_input_vector, tfidf_matrix)
    max_similarity_index = similarities.argmax()
    response = corpus[max_similarity_index]
    return response
```

Putting it All Together

Let's put everything together and create a simple chatbot that responds to predefined queries.

```
corpus = [
    'Hello',
    'How are you?',
    'What is your name?',
    'Tell me a joke',
    'Goodbye',
    'What is the weather like today?',
    'Can you recommend a good restaurant nearby?',
    'How can I contact customer support?',
    'Tell me the latest news',
    'What is the meaning of life?'
]

print("Chatbot: Hello! How can I assist you?")

# Chatbot interaction loop
```

```
while True:
    user_input = input("User: ")
    response = generate_response(user_input, corpus)
    print("Chatbot:", response)

    if user_input.lower() == 'goodbye':
        break
```

During each iteration of the loop, the chatbot takes the user's input, preprocesses it, compares it with the corpus using TF-IDF and cosine similarity, and selects the most relevant response from the corpus. The selected response is then printed as the chatbot's reply.

```
[nltk_data] Downloading package punkt to /Users/gcerri/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to /Users/gcerri/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
Chatbot: Hello! How can I assist you?
User: What's your name?
Chatbot: My name is Chatbot.
User: how is the weather today?
Chatbot: The weather is sunny today.
User: Goodbye
Chatbot: Goodbye!
```