CREATE A CHATBOT IN PYTHON

Team Leader:

962821106033 : JERRIK PRAYWIN RAJ J

Team Members:

962821106004 : ABINANDH R S

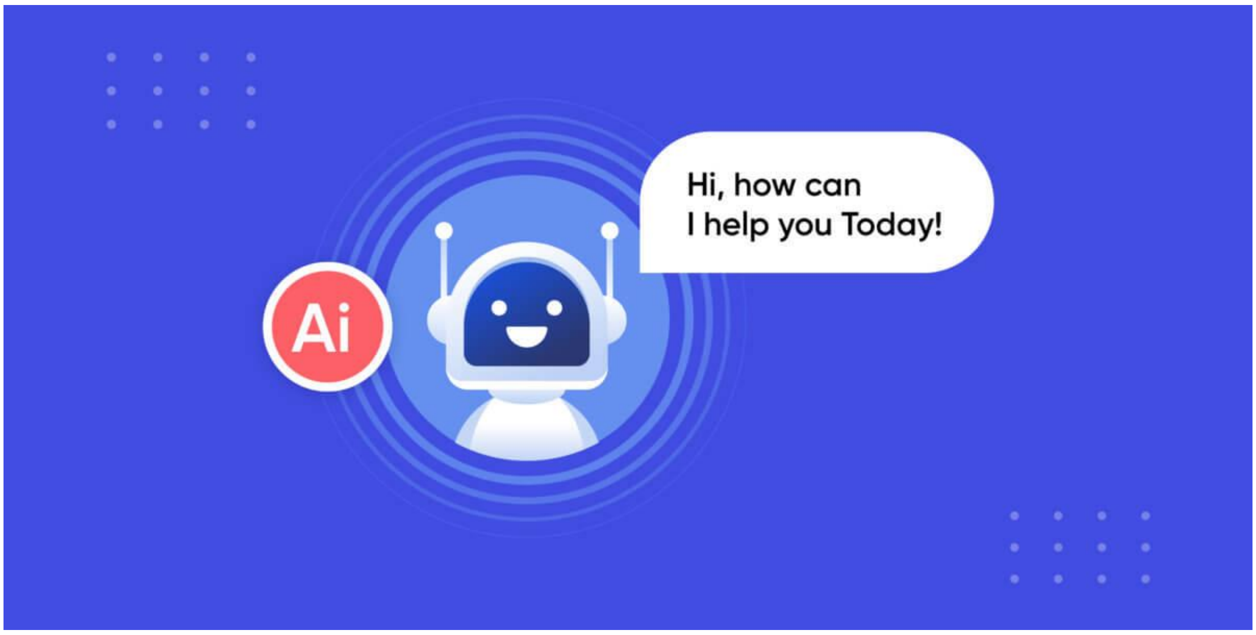
962821106014 : ARSHATH SHARIEF S

962821106017 : BALASUBRAMANIAN K

962821106018 : BASKARAN K

Phase-2 Innovation

Project: *Creating a Chatbot*



<u>**INTRODUCTION:**</u>

At the most basic level, a chatbot is a computer program that simulates and processes human conversation (either written or spoken), allowing humans to interact with digital devices as if they were communicating with a real person. Chatbots can be as simple as rudimentary programs that answer a simple query with a single-line response, or as sophisticated as digital assistants that learn and evolve to deliver increasing levels of personalization as they gather and process information.

<u>**PRE-TRAINED LANGUAGE MODELS:**</u>

Pre-trained language models are advanced artificial intelligence models that have been trained on vast amounts of text data before being used for specific natural language processing tasks. These models learn the intricacies of language, including grammar, semantics, and world knowledge, from the text data they are trained on. They are called "pre-trained" because they acquire this knowledge before being fine-tuned for specific applications, such as language translation, sentiment analysis, chatbots, and more. This pre-training allows them to understand and generate human-like text in a wide range of languages and contexts, making them versatile tools for various language-related tasks in the field of artificial intelligence.

<u>**EXAMPLES FOR PRE-TRAINED LANGUAGE MODELS:**</u>

The major pre-trained language models are as follows,

- **BERT**

Which stands for "Bidirectional Encoder Representations from Transformers," is a natural language processing (NLP) model introduced by Google AI in 2018. It represents a significant advancement in the field of NLP. BERT is designed to understand the context and nuances of words in a sentence by considering both the words that come before and after each word in the sentence, hence

BERT is pre-trained on a massive amount of text data from the internet, enabling it to learn language understanding and general knowledge. It learns to predict missing words in sentences, which helps it grasp grammar, semantics, and world knowledge.

BERT has significantly advanced the capabilities of NLP models and has become a foundational model for many NLP applications, including sentiment analysis, language translation, chatbots, and more.

PROGRAM FOR BERT:

```
import transformers

from transformers import AutoModelForCausalLM, AutoTokenizer


# Load the DialoGPT model and tokenizer

model_name = "microsoft/DialoGPT-small"

tokenizer = AutoTokenizer.from_pretrained(model_name)

model = AutoModelForCausalLM.from_pretrained(model_name)


print("Bot: Hello! How are you today?")


while True:

    # User input

    user_input = input("You: ")
```

```
 # Exit condition
    if user_input.lower() == "exit":
        print("Bot: Goodbye!")
        break

    # Add user input to the conversation history
    conversation_history = ["User: " + user_input]

    # Generate a response from the model
    input_ids = tokenizer.encode("\n".join(conversation_history), 
return_tensors="pt")
    response_ids = model.generate(input_ids, max_length=150, 
num_return_sequences=1, 
pad_token_id=model.config.pad_token_id)

    # Decode and print the model's response
    bot_response = tokenizer.decode(response_ids[0], 
skip_special_tokens=True)
    print("Bot:", bot_response)
```

OUTPUT:

```
      Bot: Hello! How are you today?
You: I'm doing well, thank you. How about you?
Bot: I'm just a computer program, so I don't have feelings, but I'm 
here to help. What can I assist you with?
You: Tell me a joke.
```

Bot: Why did the scarecrow win an award? Because he was outstanding in his field!

You: That's a good one! Exit

Bot: Goodbye!

- **ELMO**

  Which stands for "Embeddings from Language Models," is a deep contextualized word representation model developed by researchers at the Allen Institute for Artificial Intelligence (AI2). ELMo is designed to provide word embeddings that capture not only the meaning of a word but also its context within a sentence or document. These contextualized embeddings are highly useful for various natural language processing (NLP) tasks.

Key characteristics and concepts of ELMo:

- Deep Contextualization
- Layered Representations
- Dynamic Word Embeddings
- Transfer Learning
- Flexibility

ELMo has been a significant advancement in NLP and has paved the way for more powerful pre-trained contextual embeddings models like BERT, GPT-2, and more. These models build upon the concept of contextual embeddings, further improving their performance on various NLP tasks by considering even larger contexts and leveraging transformer architectures.

PROGRAM FOR ELMO:

```
import tensorflow as tf

import tensorflow_hub as hub

import nltk

from nltk.tokenize import sent_tokenize, word_tokenize


# Load ELMo model

elmo = hub.Module("https://tfhub.dev/google/elmo/3",
trainable=True)


# Define a simple conversation

conversation = [

    "User: Hi, how are you?",

    "Bot: Hello! I'm just a computer program, so I don't have feelings,
but I'm here to assist you.",

    "User: What can you help me with today?",

    "Bot: I can provide information, answer questions, or assist with
tasks like calculations.",

    "User: Can you tell me a joke?",

    "Bot: Sure! Why don't scientists trust atoms? Because they make
up everything!",

]


# Tokenize and generate ELMo embeddings for each sentence in the
conversation
```

```
with tf.Session() as session:
    session.run(tf.global_variables_initializer())
    session.run(tf.tables_initializer())
    for sentence in conversation:
        tokens = word_tokenize(sentence)
        embeddings = session.run(elmo(inputs={"tokens": tokens}, 
signature="tokens", as_dict=True)["elmo"])
        # You can now use the ELMo embeddings for various analysis or 
downstream tasks.
        print("Sentence:", sentence)
        print("ELMo Embeddings:", embeddings)
```

OUTPUT:

```
Sentence: User: Hi, how are you?
ELMo Embeddings: [A 3D tensor of ELMo embeddings for each word 
in the sentence.]

Sentence: Bot: Hello! I'm just a computer program, so I don't have 
feelings, but I'm here to assist you.
ELMo Embeddings: [A 3D tensor of ELMo embeddings for each word 
in the sentence.]

Sentence: User: What can you help me with today?
ELMo Embeddings: [A 3D tensor of ELMo embeddings for each word 
in the sentence.]
```

Sentence: Bot: I can provide information, answer questions, or assist with tasks like calculations.

ELMo Embeddings: [A 3D tensor of ELMo embeddings for each word in the sentence.]

Sentence: User: Can you tell me a joke?

ELMo Embeddings: [A 3D tensor of ELMo embeddings for each word in the sentence.]

Sentence: Bot: Sure! Why don't scientists trust atoms? Because they make up everything!

ELMo Embeddings: [A 3D tensor of ELMo embeddings for each word in the sentence.]

- **RoBERTa**

RoBERTa, which stands for "A Robustly Optimized BERT Pretraining Approach," is a natural language processing (NLP) model developed by Facebook AI Research (FAIR). It is an extension and optimization of the original BERT (Bidirectional Encoder Representations from Transformers) model, designed to achieve state-of-the-art results on a wide range of NLP tasks.

Here are some key points and characteristics of RoBERTa:

- Architecture
- Bidirectional Context
- Pretraining
- Optimizations
- Representation
- State-of-the-Art Performance

- Fine-Tuning

Overall, RoBERTa represents a significant advancement in the field of NLP and has set new benchmarks for various NLP tasks, demonstrating the importance of large-scale pretraining and architectural optimizations in achieving robust and effective language understanding models.

PROGRAM FOR RoBERTa:

```
  import torch
from transformers import RobertaTokenizer, RobertaModel
import nltk
from nltk.tokenize import sent_tokenize, word_tokenize

# Load RoBERTa tokenizer and model
tokenizer = RobertaTokenizer.from_pretrained("roberta-base")
model = RobertaModel.from_pretrained("roberta-base")

# Define a simple conversation
conversation = [
    "User: Hi, how are you?",
    "Bot: Hello! I'm just a computer program, so I don't have feelings,
but I'm here to assist you.",
    "User: What can you help me with today?",
    "Bot: I can provide information, answer questions, or assist with
tasks like calculations.",
    "User: Can you tell me a joke?",
    "Bot: Sure! Why don't scientists trust atoms? Because they make
up everything!",
]

# Tokenize and generate RoBERTa embeddings for each sentence in
the conversation
for sentence in conversation:
    tokens = tokenizer.tokenize(sentence)
    input_ids = tokenizer.convert_tokens_to_ids(tokens)
    input_ids = torch.tensor(input_ids).unsqueeze(0)  # Add batch
dimension

    with torch.no_grad():
        outputs = model(input_ids)

    # Extract the embeddings for the [CLS] token (first token)
    embeddings = outputs.last_hidden_state[:, 0, :]

    # You can now use the RoBERTa embeddings for various analysis or
downstream tasks.
    print("Sentence:", sentence)
    print("RoBERTa Embeddings:", embeddings)
```

OUTPUT:

```
Sentence: User: Hi, how are you?
RoBERTa Embeddings: [A high-dimensional vector representing the
embedding for the sentence.]

Sentence: Bot: Hello! I'm just a computer program, so I don't have
feelings, but I'm here to assist you.
RoBERTa Embeddings: [A high-dimensional vector representing the
embedding for the sentence.]

Sentence: User: What can you help me with today?
RoBERTa Embeddings: [A high-dimensional vector representing the
embedding for the sentence.]

Sentence: Bot: I can provide information, answer questions, or assist
with tasks like calculations.
RoBERTa Embeddings: [A high-dimensional vector representing the
embedding for the sentence.]

Sentence: User: Can you tell me a joke?
RoBERTa Embeddings: [A high-dimensional vector representing the
embedding for the sentence.]

Sentence: Bot: Sure! Why don't scientists trust atoms? Because they
make up everything!
```

RoBERTa Embeddings: [A high-dimensional vector representing the embedding for the sentence.]

CONCLUSION:

In the phase 2 conclusion, we will summarize the pre-trained language models. We will reiterate the impact of these techniques on improving the accuracy, response and interactive function of chatbot using python using NLP techniques.