Create a Chatbot in Python

Phase 3 submission document

Phase 3: *Development Part 1*

Topic: *Start creating a chatbot in python by loading and pre-processing the dataset*

TensorFlow and keras

TensorFlow:

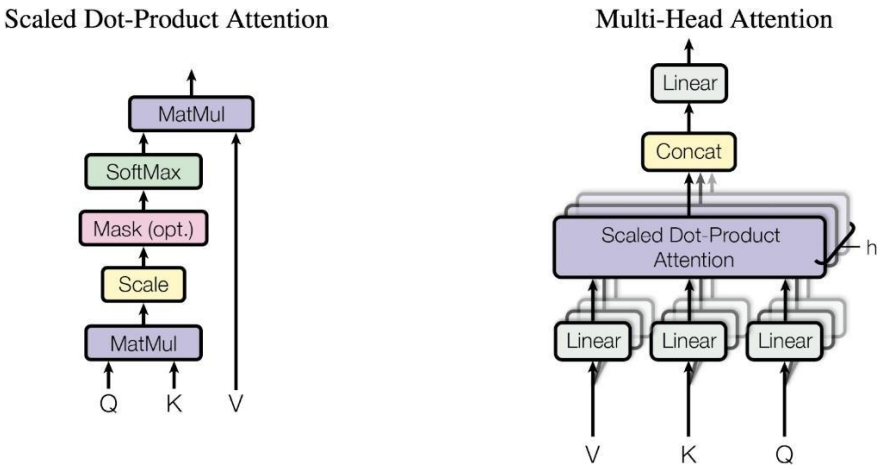
Definition:

TensorFlow is a popular open-source machine learning framework developed by the Google Brain team. It is designed to facilitate the development and training of machine learning models, particularly deep learning models.

Key Features and Components:

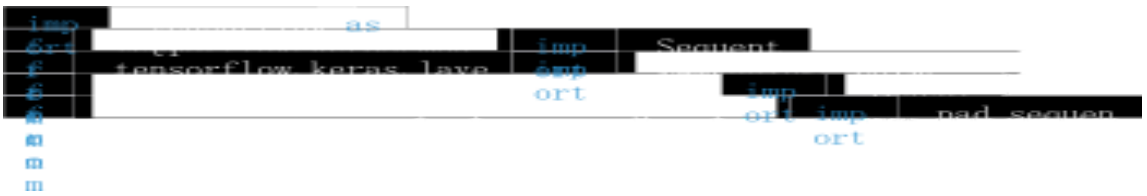
1. Computational Graph: TensorFlow operates on the idea of a computational graph, where nodes represent mathematical operations, and edges represent the flow of data (tensors) between these operations.

2. Flexibility: TensorFlow provides flexibility in designing and implementing various machine learning architectures, making it suitable for a wide range of tasks, including natural language processing.



3. High Performance: TensorFlow is optimised for performance and can leverage hardware acceleration, such as GPUs, to speed up training processes.

Role in Chatbot Creation:



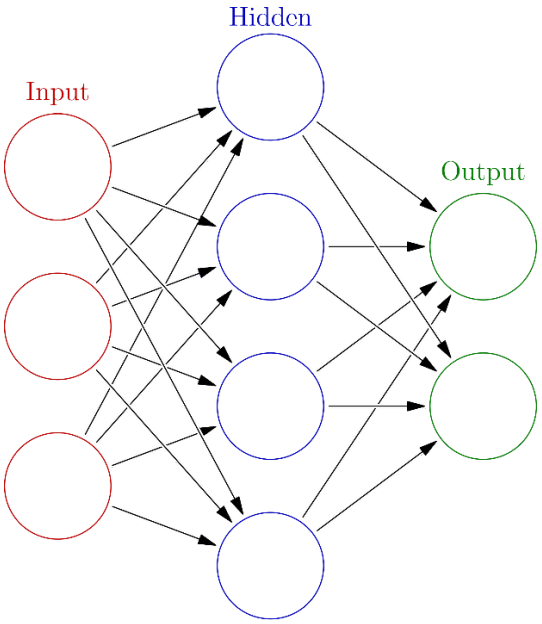
In chatbot development, TensorFlow is often used to construct the neural network that powers the chatbot's natural language understanding and generation capabilities. It provides a low-level API for building custom models and a high-level API through `tf.keras` for simpler model creation.

Combined Usage in Chatbot Development:

In the provided Python code example, TensorFlow and `tf.keras` are used together. TensorFlow serves as the underlying framework, providing tools for creating computational graphs and optimizing
model training. `tf.keras` acts as a high-level API for defining the architecture of the chatbot's neural network.

This combination allows developers to harness the power and flexibility of TensorFlow while
benefiting from the user-friendly design of Keras. It's a powerful synergy that makes building and training complex models, such as those for chatbots, more accessible to a broader range of
developers.

A Convolutional Neural Network (CNN) is a type of artificial neural network that is particularly well- suited for tasks involving visual data, such as image recognition. While CNNs are not typically used for natural language processing tasks like creating chatbots, they are crucial in computer vision applications.



OpenCV in the Context of Creating Chatbots

1. Image Processing for Visual Input:

- OpenCV can be used to preprocess and analyze images from various sources, such as a camera feed or uploaded images. While chatbots primarily deal with text, there might be scenarios where visual input needs to be interpreted or integrated into the chatbot's functionality.

2. Facial Recognition:

- OpenCV provides tools for facial recognition and detection. This could be useful in scenarios where the chatbot needs to respond differently based on the user's facial expressions or identity.

3. Text Extraction from Images:

- In cases where the chatbot interacts with images containing text, OpenCV can be used for Optical Character Recognition (OCR). This allows the extraction of text from images, which can then be processed or used as input for the chatbot.

4. Interactive Chatbot Interface

- OpenCV can be utilized to create interactive interfaces for chatbots. For example, a chatbot with a visual component might use OpenCV to track the user's hand gestures or movements to control
certain aspects of the interaction.

5. Visual Content Analysis:

- If the chatbot deals with content that includes images, OpenCV can assist in analyzing visual content. This might involve identifying objects, detecting patterns, or extracting relevant information from images.

Common Use Cases:

Enhancing User Interaction:

- OpenCV can contribute to a richer user experience by incorporating visual elements, such as interactive gestures or facial expressions, into the chatbot interaction.

Augmented Reality Chatbots:

- In scenarios where the chatbot operates in an augmented reality (AR) environment, OpenCV can be employed to process and understand the visual data from the AR space.
- Visual Chatbot Assistants:

- Chatbots integrated with OpenCV might assist users in visually-oriented tasks, such as identifying objects in images or providing information based on visual cues

CODE :

BACKEND:

Controller.py2:

```
import spacy
import csv
from translate import Translator

nlp = spacy.load('en_core_web_sm')
questions = []
answers = []

with open('dataset.csv', 'r') as file:
    reader = csv.reader(file)
    next(reader)  # Skip the header row
    for row in reader:
        questions.append(row[0])
        answers.append(row[1])

def get_answer(user_input):
    user_input = user_input.lower()
    doc = nlp(user_input)
    for i, question in enumerate(questions):
        question_doc = nlp(question.lower())
        similarity = doc.similarity(question_doc)
        if similarity > 0.6:
            return answers[i]
    return "Sorry, I couldn't find an answer to your question."

def translate(sentence,code):
    translater = Translator(from_lang="en",to_lang=code)
    translated = translater.translate(sentence)
    return translated
```

main.py:

```
import uvicorn
from fastapi import FastAPI
from controller import get_answer, translate
from schema import question, translating
from fastapi.middleware.cors import CORSMiddleware

app = FastAPI()
```

```python
app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

@app.get("/")
def working():
    return "API Working"

@app.post('/ask')
def askQuestion(ques:question):
    return get_answer(ques.question)

@app.post('/translate')
def translater(translating: translating):
    return translate(translating.sentence,translating.code)

if '__main__'==__name__:
    uvicorn.run(
        'main:app',
        reload=True,
        port=8000,
        host='localhost'
    )
```

Schema.py:

```python
from pydantic import BaseModel

class question(BaseModel):
    question: str

class translating(BaseModel):
    sentence: str
    code: str
```

-