Experiment-2

Simulate of a Bunch of Helium Molecules

Date: 14/8/24

AIM

Simulate of a Bunch of Helium Molecules.

PROCEDURE

we'll consider the following assumptions and simplifications:

Step-1: Helium atoms are treated as hard spheres.

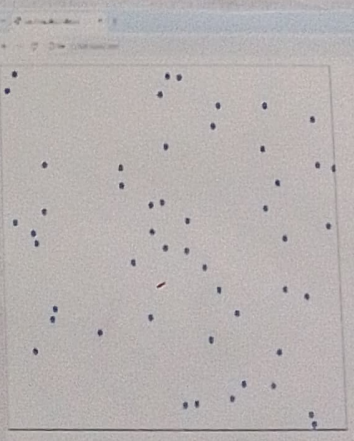
Step-2: Interactions between atoms are modeled using a simple Lennard-Jones potential.

Step-3: Periodic boundary conditions are used to simulate an infinite system.

SOURCE CODE

Prompt 1:
create a physics simulation in javascript of container full of gas molecules. The molecules should be simple diatomic helium gas molecules. Also in addition to collision physics also add corrections for vanderwaals forces and apply distortion to the mechanics of the molecules based on these forces.

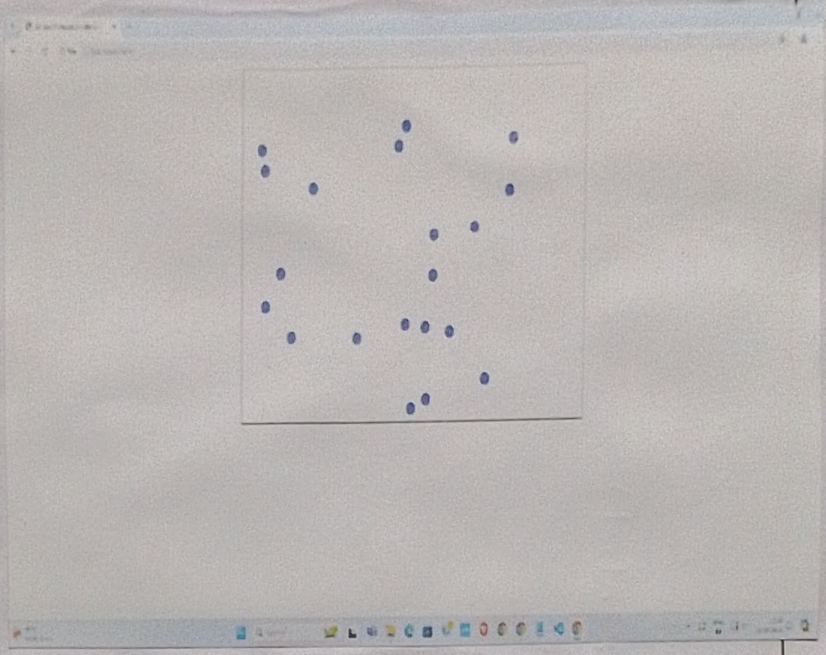
output:-



Prompt 2:

Create a simple 2D demonstration of the above in javascript to the level of complexity that you can code.

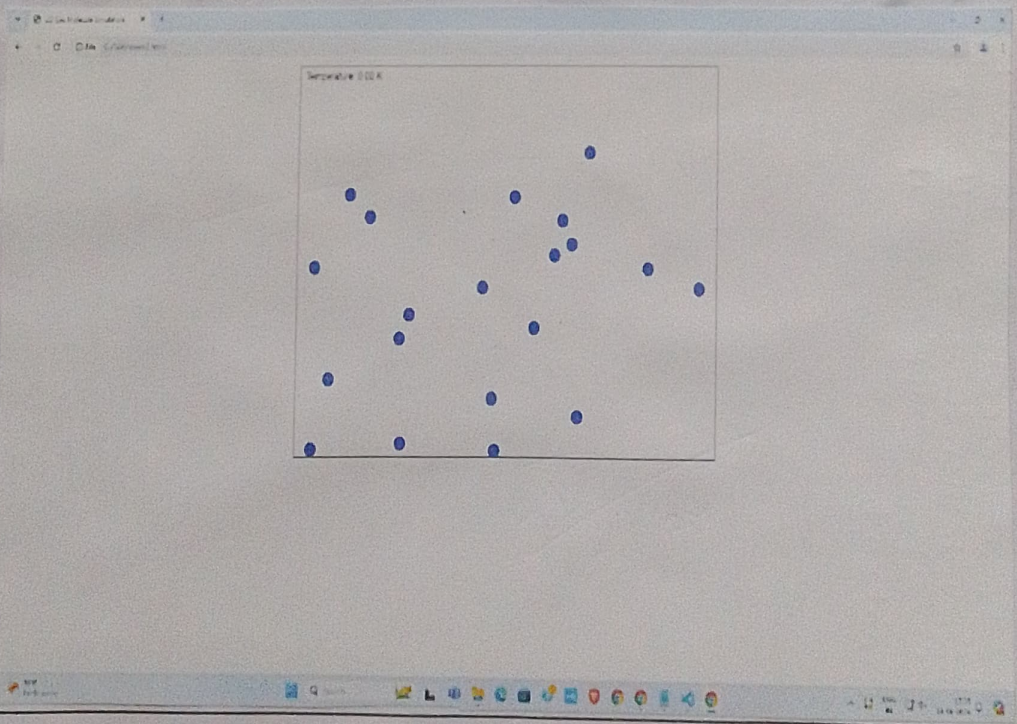
output:-



Prompt 3:

Using the mass of the helium nucleus, and the speeds of the gas molecules, have label that outputs the temperature of the box at any instant. Use statistical mechanics to calculate the temperature.

output:-



Prompt 4:

Update above code to add the label in html below the gas container. To the right of the gas container have a small box that explains the kinetic theory of gases and have the relavant equation below it.

Final Code (index.html)

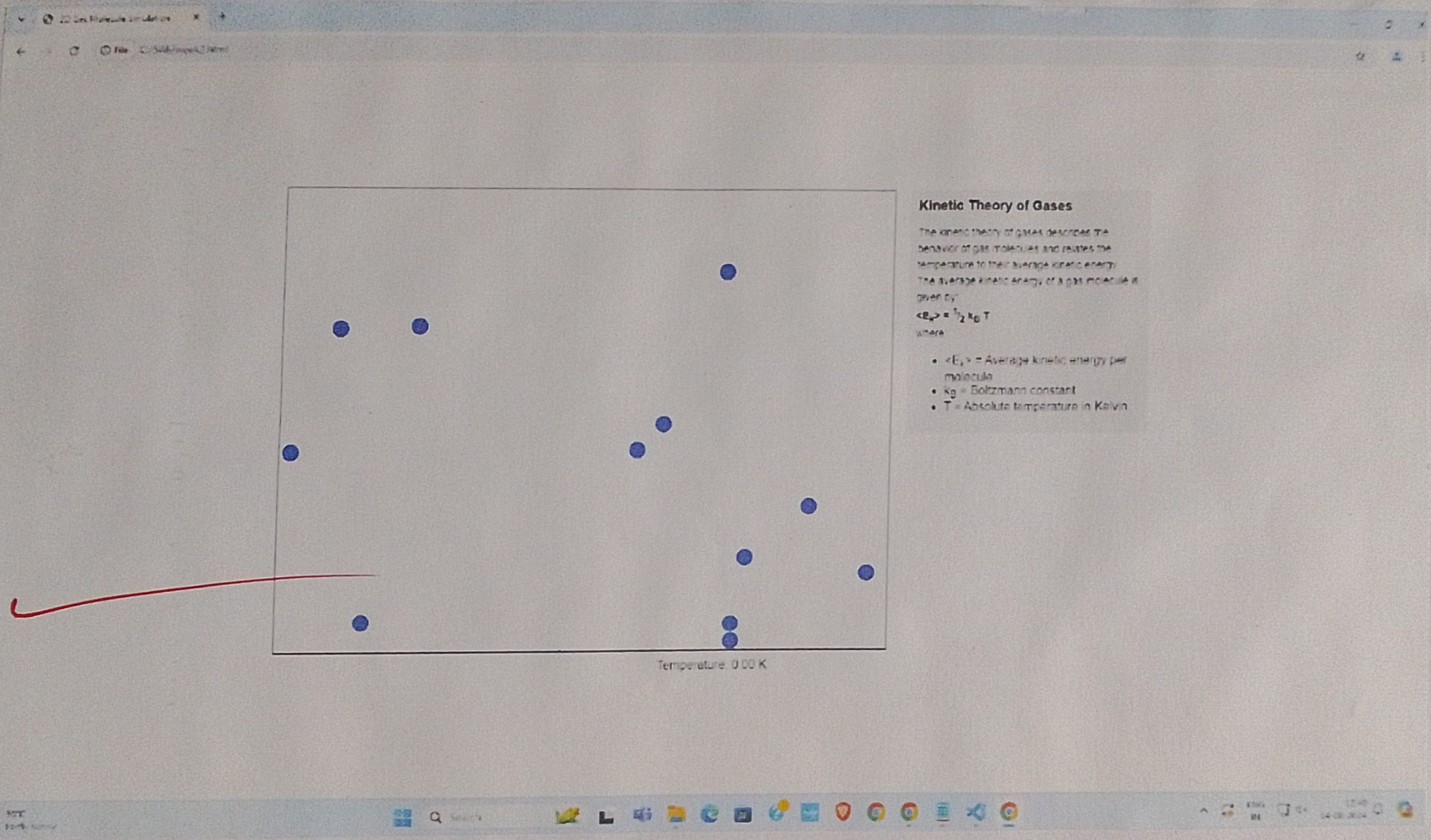
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>2D Gas Molecule Simulation</title>
    <style>
        body {
            display: flex;
            justify-content: center;
            align-items: center;
            flex-direction: column;
            height: 100vh;
            margin: 0;
            font-family: Arial, sans-serif;
        }
        .container {
            display: flex;
            flex-direction: column;
            align-items: center;
        }
        #simulationWrapper {
            display: flex;
            flex-direction: row;
            align-items: flex-start;
        }
        canvas {
            border: 1px solid black;
        }
        #infoBox {
            margin-left: 20px;
            padding: 10px;
            width: 300px;
            background-color: #f0f0f0;
            border-radius: 5px;
        }
        #infoBox h2 {
            font-size: 18px;
            margin-top: 0;
        }
        #infoBox p {
            font-size: 14px;
            margin: 0;
            line-height: 1.5;
        }
        #temperatureLabel {
            margin-top: 10px;
            font-size: 16px;
        }
    </style>
</head>
```

```
<body>
    <div class="container">
        <div id="simulationWrapper">
            <canvas id="simulationCanvas" width="800" height="600"></canvas>
            <div id="infoBox">
                <h2>Kinetic Theory of Gases</h2>
    <p>The kinetic theory of gases describes the behavior of gas molecules and relates the temperature to their average kinetic energy.</p>
                <p>The average kinetic energy of a gas molecule is given by:</p>
                <p><strong>&#60;E<sub>k</sub>&#62; = <sup>3</sup>/<sub>2</sub> k<sub>B</sub> T</strong></p>
                <p>where:</p>
                <ul>
                    <li>&#60;E<sub>k</sub>&#62; = Average kinetic energy per molecule</li>
                    <li>k<sub>B</sub> = Boltzmann constant</li>
                    <li>T = Absolute temperature in Kelvin</li>
                </ul>
            </div>
        </div>
        <div id="temperatureLabel">Temperature: N/A K</div>
    </div>
    <script>
        const canvas = document.getElementById('simulationCanvas');
        const ctx = canvas.getContext('2d');

        const numMolecules = 12;  // Number of molecules
        const moleculeRadius = 10;
        const containerWidth = canvas.width;
        const containerHeight = canvas.height;

        // Constants
        const boltzmannConstant = 1.38e-23;  // J/K
        const heliumMass = 6.64e-27;  // kg (mass of helium-4 nucleus)

        const molecules = [];

        // Molecule class
        class Molecule {
            constructor(x, y, vx, vy) {
                this.x = x;
                this.y = y;
                this.vx = vx;
                this.vy = vy;
            }

            updatePosition() {
                this.x += this.vx;
                this.y += this.vy;

                // Wall collision detection
                if (this.x - moleculeRadius < 0 || this.x + moleculeRadius > containerWidth) {
```

```
            }
            if (this.y - moleculeRadius < 0 || this.y + moleculeRadius > containerHeight) {
                this.vy *= -1;
            }
        }

        applyVanDerWaalsForce(other) {
            const dx = other.x - this.x;
            const dy = other.y - this.y;
            const distance = Math.sqrt(dx * dx + dy * dy);

            if (distance < 2 * moleculeRadius && distance > 0) {
                const force = 0.01 / distance; // Simplified attraction/repulsion force
                const ax = force * dx / distance;
                const ay = force * dy / distance;

                this.vx -= ax;
                this.vy -= ay;
                other.vx += ax;
                other.vy += ay;
            }
        }

        draw() {
            ctx.beginPath();
            ctx.arc(this.x, this.y, moleculeRadius, 0, Math.PI * 2);
            ctx.fillStyle = 'blue';
            ctx.fill();
            ctx.stroke();
        }
    }

    // Initialize molecules with random positions and velocities
    for (let i = 0; i < numMolecules; i++) {
        const x = Math.random() * (containerWidth - 2 * moleculeRadius) + moleculeRadius;
        const y = Math.random() * (containerHeight - 2 * moleculeRadius) + moleculeRadius;
        const vx = (Math.random() - 0.5) * 2;
        const vy = (Math.random() - 0.5) * 2;
        molecules.push(new Molecule(x, y, vx, vy));
    }

    function calculateTemperature() {
        let sumV2 = 0;

        // Calculate the average of v^2
        molecules.forEach(molecule => {
            const v2 = molecule.vx * molecule.vx + molecule.vy * molecule.vy;
            sumV2 += v2;
        });
```

```
            const avgV2 = sumV2 / numMolecules;
            // Calculate temperature
            return (heliumMass * avgV2) / (3 * boltzmannConstant);
        }

        // Simulation loop
        function updateSimulation() {
            ctx.clearRect(0, 0, containerWidth, containerHeight);

            for (let i = 0; i < molecules.length; i++) {
                for (let j = i + 1; j < molecules.length; j++) {
                    molecules[i].applyVanDerWaalsForce(molecules[j]);
                }
            }

            molecules.forEach(molecule => {
                molecule.updatePosition();
                molecule.draw();
            });

            // Calculate and display temperature
            const temperature = calculateTemperature();
            document.getElementById('temperatureLabel').innerText = `Temperature: ${temperature.toFixed(2)} K`;

            requestAnimationFrame(updateSimulation);
        }

        // Start the simulation
        updateSimulation();
    </script>
</body>
</html>
```

output :-


Temperature: 0.00 K
Kinetic Theory of Gases
The kinetic theory of gases describes the behavior of gas molecules and relates the temperature to their average kinetic energy. The average kinetic energy of a gas molecule is given by:
$\langle E_k \rangle = \frac{1}{2} k_B T$
where
• $\langle E_k \rangle$ = Average kinetic energy per molecule
• k_B = Boltzmann constant
• T = Absolute temperature in Kelvin

VIVA QUESTIONS

1. Can you briefly describe the objective of your simulation?

Ans. The simulation visualizes the movement and collisions of 12 diatomic helium molecules in a 2D container, with real-time temperature calculation based on kinetic theory. It also provides an educational overview of the relevant formula.

2. What assumptions did you make in your simulation?

Ans. The simulation assumes ideal gas behavior with elastic collisions and simplified Vanderwaals forces in a 2D plane, using diatomic molecules (though helium is monatomic) and a basic model for constant temp. calculation. These simplifications aid in educational understanding.

3. Can you explain the Lennard-Jones potential and its significance in your simulation?

Ans. The Lennard-Jones potential models molecular interactions by combining attractive and repulsive forces, preventing collapse at short distances while pulling molecules together at longer distances. In simulation, it offers a more realistic interaction model, though a simplified version is used for easier implementation.

4. How did you implement periodic boundary conditions in your simulation?

Ans. The simulation uses periodic boundary conditions, where molecules exiting one side of the container reappear on the opposite side, maintaining their other coordinate. This creates the effect of an infinite, repeating system, ensuring consistent molecule interactions within the visible area.

5. What kind of analyses can you perform on the simulation data?

Ans. The simulation allows for analyses including temperature monitoring, velocity distribution, collision frequency, molecular trajectories and pressure estimation. These analyses help in understanding gas behavior, verifying theoretical models, and exploring parameter effects on the system.

6. How do you ensure that the simulation results are physically meaningful and accurate?

Ans. To ensure accurate simulation results, use correct physical equations, validate against theory or data, implement proper boundary conditions, select an appropriate time step, and fine-tune parameters to reflect realistic molecular behavior. These practices align the simulation with physical principles.

Experiment-3

Implement Natural Language Processing in Multi Sentence Conversation

Date: 21/8/24

AIM

Implement natural language processing in multi sentence conversation.

PROCEDURE

Step-1: Text Preprocessing

Step-2: Feature Extraction

Step-3: Applying NLP Models

Step-4: Text Classification

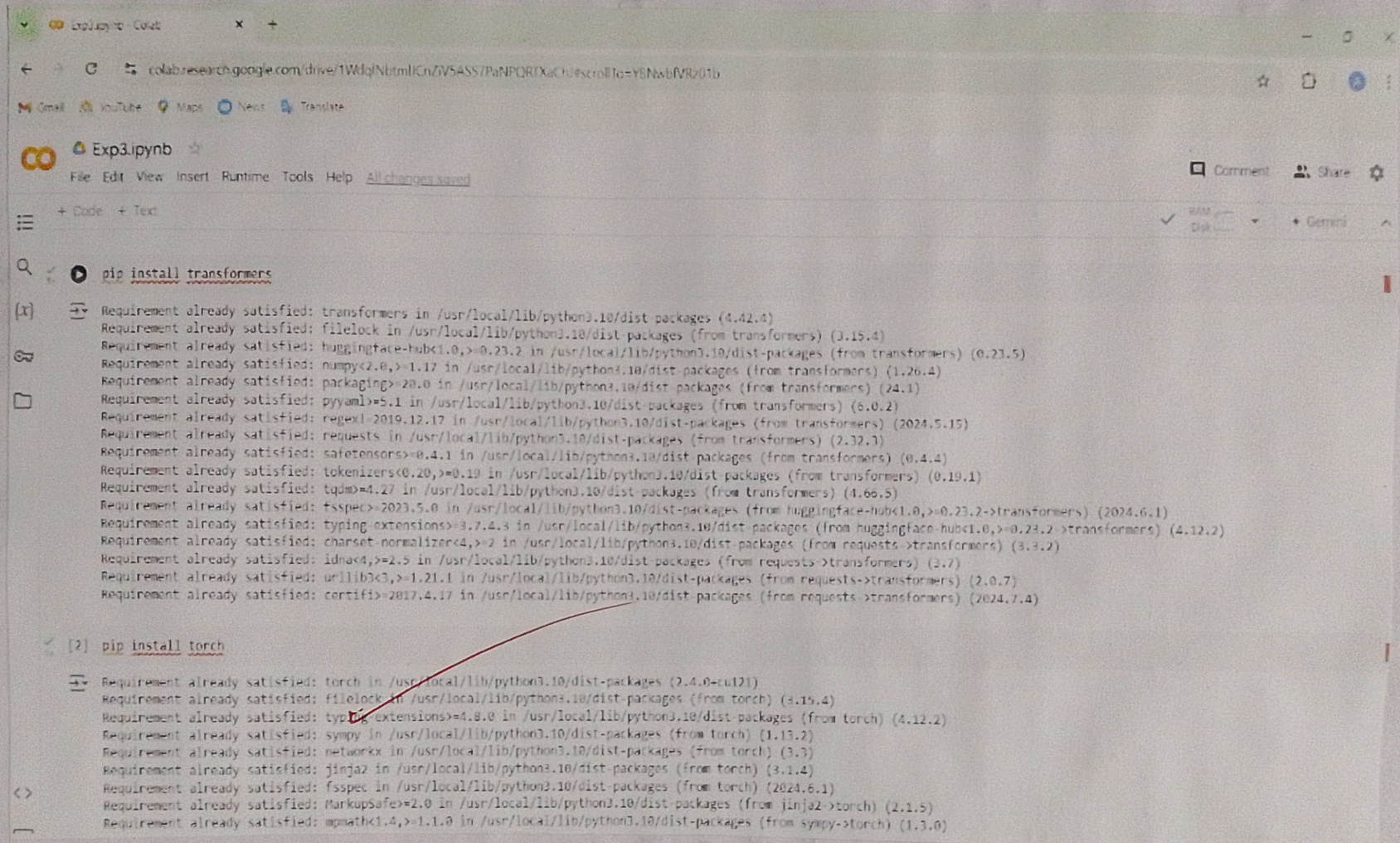
SOURCE CODE

step1: open browser > search openAI > click on try chatgpt > Login using your credentials.

step2: Now login and generate the prompt mentioning your requirements of implementing NLP in multi sentence conversation between you and a chatbot.

Prompt:

Create a Python script using the transformers library to build a simple chatbot powered by GPT-2. Begin by importing GPT2Tokenizer, GPT2LMHeadModel, and pipeline, then load the pre-trained GPT-2 model (gpt2-medium) along with its tokenizer. Initialize a text generation pipeline using these components. Define a ChatBot class that manages the conversation context and generates responses based on user input, with a method to update the context and produce coherent replies. Finally, instantiate the chatbot and simulate a multi-sentence conversation, printing both user inputs and the bot's responses to demonstrate its functionality.


Exp3.ipynb - Colab
colab.research.google.com/drive/1WdqlNbtmlICnZV5ASS7PaNPQRTXaCh#scrollTo=Y8NwbfVR201b
Gmail YouTube Maps News Translate
Exp3.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Gemini
pip install transformers
Requirement already satisfied: transformers in /usr/local/lib/python3.10/dist-packages (4.42.4)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from transformers) (3.15.4)
Requirement already satisfied: huggingface-hub<1.0,>=0.23.2 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.23.5)
Requirement already satisfied: numpy<2.0,>=1.17 in /usr/local/lib/python3.10/dist-packages (from transformers) (1.26.4)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from transformers) (24.1)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from transformers) (6.0.2)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-packages (from transformers) (2024.5.15)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from transformers) (2.32.3)
Requirement already satisfied: safetensors>=0.4.1 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.4.4)
Requirement already satisfied: tokenizers<0.20,>=0.19 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.19.1)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dist-packages (from transformers) (4.66.5)
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.23.2->transformers) (2024.6.1)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.23.2->transformers) (4.12.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (2024.7.4)
[2] pip install torch
Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-packages (2.4.0+cu121)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch) (3.15.4)
Requirement already satisfied: typing-extensions>=4.8.0 in /usr/local/lib/python3.10/dist-packages (from torch) (4.12.2)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch) (1.13.2)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch) (3.3)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch) (3.1.4)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from torch) (2024.6.1)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->torch) (2.1.5)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from sympy->torch) (1.3.0)

Exp3.ipynb - Colab

colab.research.google.com/drive/1WdqiNbtmlJCnZiV5AS57PaNPQRTXaChi#scrollTo=Y6NwbfVRz01b

Gmail YouTube Maps News Translate

Exp3.ipynb

File Edit View Insert Runtime Tools Help All changes saved

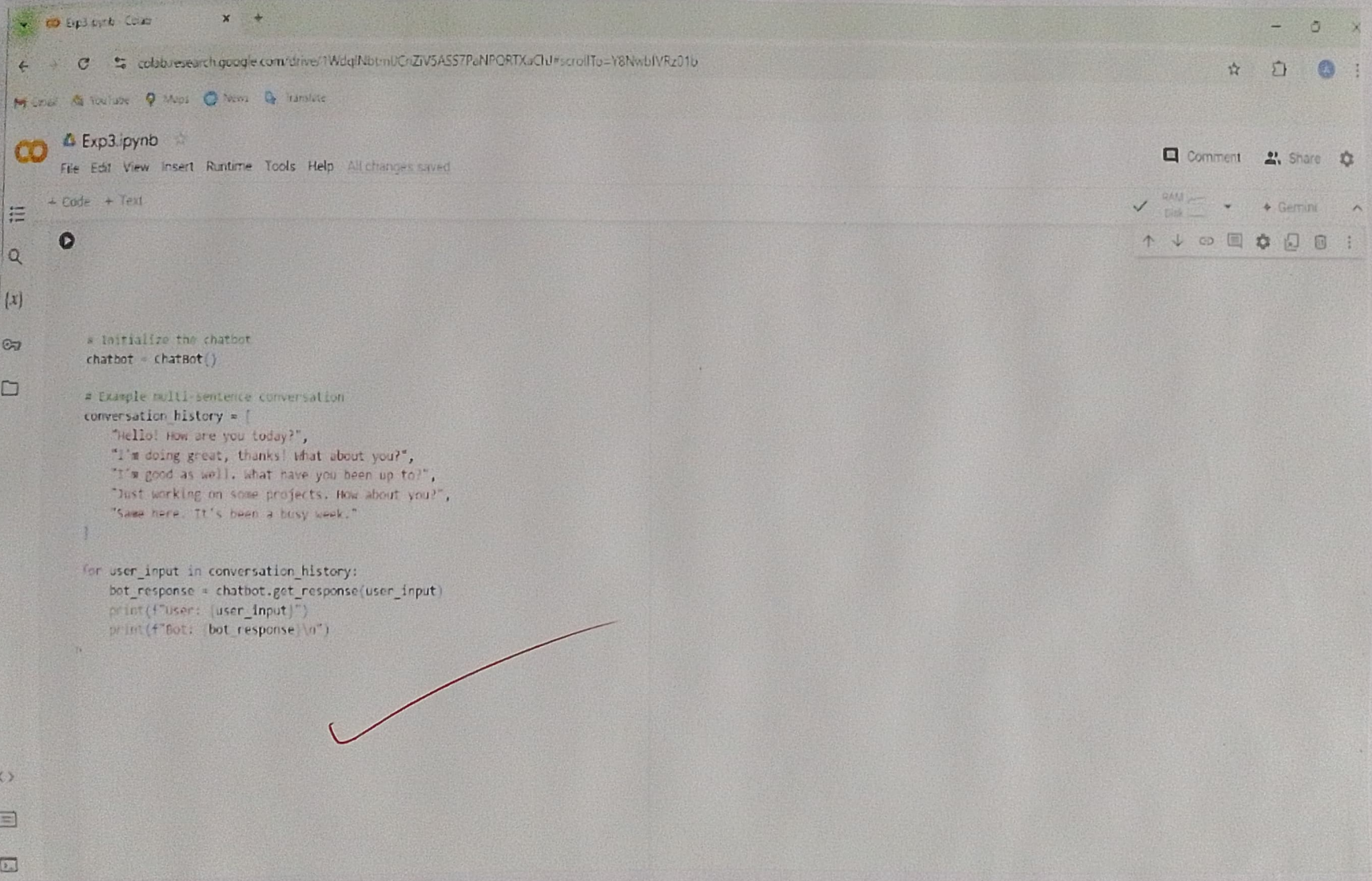
Comment Share

+ Code + Text

RAM Disk

Gemini

```
from transformers import GPT2Tokenizer, GPT2LMHeadModel, pipeline


    # Load pre-trained model and tokenizer
    model_name = 'gpt2-medium'
    tokenizer = GPT2Tokenizer.from_pretrained(model_name)
    model = GPT2LMHeadModel.from_pretrained(model_name)

    # Initialize the text generation pipeline
    text_generation_pipeline = pipeline('text-generation', model=model, tokenizer=tokenizer)


    class ChatBot:
        def __init__(self, model_name='gpt2-medium'):
            self.tokenizer = GPT2Tokenizer.from_pretrained(model_name)
            self.model = GPT2LMHeadModel.from_pretrained(model_name)
            self.pipeline = pipeline('text-generation', model=self.model, tokenizer=self.tokenizer)
            self.context = ""

        def get_response(self, user_input):
            # Update context
            self.context += f"User: {user_input}\nBot: "

            # Generate response
            response = self.pipeline(self.context, max_length=500, pad_token_id=self.tokenizer.eos_token_id, num_return_sequences=1)

            # Extract and update context with the response
            bot_response = response[0]['generated_text'].split("Bot: ")[-1].split("User: ")[0].strip()
            self.context += f"{bot_response}\n"
            return bot_response
```


Exp3.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
Gemini
# Initialize the chatbot
chatbot = ChatBot()
# Example multi-sentence conversation
conversation_history = [
"Hello! How are you today?",
"I'm doing great, thanks! What about you?",
"I'm good as well. What have you been up to?",
"Just working on some projects. How about you?",
"Same here. It's been a busy week."
]
for user_input in conversation_history:
bot_response = chatbot.get_response(user_input)
print(f"User: {user_input}")
print(f"Bot: {bot_response}\n")

Exp3.ipynb - Colab

colab.research.google.com/drive/1WdqlNbImlKnZiV5AS57PaNPQRTXaChJ#scrollTo=Y8NwbfVRz01b

Gmail YouTube Maps News Translate

Exp3.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Gemini

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:89: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as secret in your Google Colab and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
tokenizer_config.json: 100% 26.0/26.0 [00:00<00:00, 651B/s]
vocab.json: 100% 1.04M/1.04M [00:00<00:00, 3.91MB/s]
merges.txt: 100% 456k/456k [00:00<00:00, 6.56MB/s]
tokenizer.json: 100% 1.36M/1.36M [00:00<00:00, 7.83MB/s]
config.json: 100% 718/718 [00:00<00:00, 6.44kB/s]
model.safetensors: 100% 1.52G/1.52G [00:17<00:00, 130MB/s]
generation_config.json: 100% 124/124 [00:00<00:00, 7.49kB/s]
Truncation was not explicitly activated but `max_length` is provided a specific value, please use `truncation=True` to explicitly truncate examples to max length. Defaulting to 'longe
User: Hello! How are you today?
Bot: We just have some people that do it very efficiently.
Anonymous: Right, it doesn't matter if I was only working and I wouldn't say the same thing on a holiday and I'm not here with my family so

User: I'm doing great, thanks! What about you?
Bot: Of course, I really want to work for you
Anonymous: Do you have any tips for those people who may not make

User: I'm good as well. What have you been up to?
Bot: No kidding, I hope you understand everything. As soon as I come back, I will come back and say your name, thank you very much
Anonymous: How did you end up coming back?

User: Just working on some projects. How about you?
Bot: Well, you are

User: Same here. It's been a busy week.
Bot: That's all.
```

VIVA QUESTIONS

1. Can you explain the main steps involved in preprocessing text for NLP tasks?

Ans. The main steps for text preprocessing in NLP include tokenization, lowercasing, removing stopwords, stemming/lemmatization and handling special characters or punctuation.

2. What is the purpose of using the TF-IDF method in NLP?

Ans. TF-IDF (Term Frequency - Inverse Document Frequency) is used to assess the importance of a word in a document relative to a collection of documents, helping to identify keywords.

3. How does the Bag-of-Words model differ from word embeddings?

Ans. The Bag of words model represents text as a sparse vector of word counts without considering context, while word embeddings encode words as dense vectors that capture semantic relationships.

4. What is the importance of Named Entity Recognition (NER) in NLP?

Ans. NER identifies and categorizes key entities in text, enabling better understanding and extraction of relevant information.

5. What challenges might you encounter when processing multi-sentence conversations, and how can you address them?

Ans. challenges include handling context, conference resolution and ambiguity. These can be addressed by using advanced models like transformers for context, conference resolution algorithms, and incorporating domain-specific knowledge.

2/2/18