**Program-01: Marketer to Machine: Develop a ML model for Smart email Compose. Smart email compose finishes sentences for you by predicting what word or words user will type next.**

Code :

pip install transformers torch

from transformers import GPT2LMHeadModel, GPT2Tokenizer

import torch

# Load upgraded model and tokenizer

print("Loading GPT-2 Medium model... (first time only)")

model\_name = "gpt2-medium"  # More accurate than gpt2

model = GPT2LMHeadModel.from\_pretrained(model\_name)

tokenizer = GPT2Tokenizer.from\_pretrained(model\_name)

def generate\_suggestions(prompt, max\_new\_tokens=15, num\_suggestions=5):

    input\_ids = tokenizer.encode(prompt, return\_tensors='pt')

    outputs = model.generate(

        input\_ids,

        max\_length=input\_ids.shape[1] + max\_new\_tokens,

        num\_return\_sequences=num\_suggestions,

        do\_sample=True,

        top\_k=40,               # Sample from top 40 words

        top\_p=0.92,             # Nucleus sampling

        temperature=0.7,        # Lower temp for focused output

        no\_repeat\_ngram\_size=2,

        pad\_token\_id=tokenizer.eos\_token\_id,

    )

    suggestions = set()

    for output in outputs:

        full\_text = tokenizer.decode(output, skip\_special\_tokens=True)

        if prompt in full\_text:

            suggestion = full\_text[len(prompt):].strip()

            suggestion = suggestion.split('.')[0].strip()  # get first clause

            if suggestion and len(suggestion.split()) > 1:

                suggestions.add(suggestion)

    return list(suggestions)[:num\_suggestions]

def main():

    user\_input = input("Start typing your email: ").strip()

    while True:

        suggestions = generate\_suggestions(user\_input)

        if not suggestions:

            print("No suggestions found. Try again.")

            break

        print("\nSuggestions:")

        for idx, suggestion in enumerate(suggestions, 1):

            print(f"{idx}. {suggestion}")

        choice = input("\nChoose a suggestion number to append (or press Enter to exit): ").strip()

        if choice.isdigit() and 1 <= int(choice) <= len(suggestions):

            selected = suggestions[int(choice) - 1]

            user\_input += " " + selected

            print(f"\nUpdated text: {user\_input}")

        else:

            break

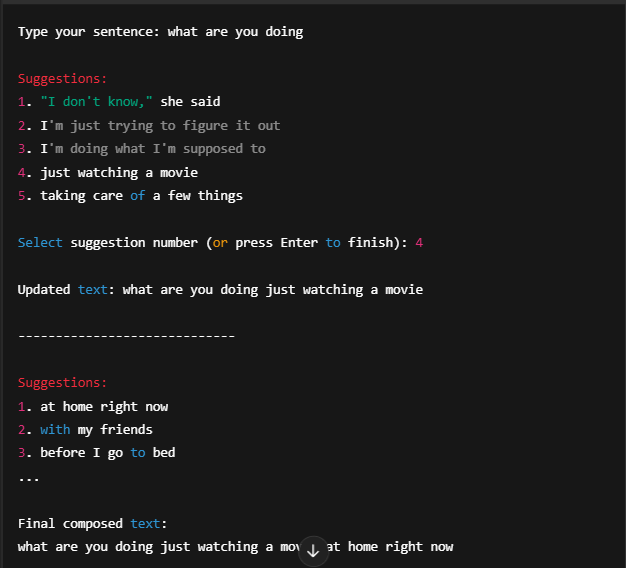
        print("\n----------------------------")

    print(f"\nFinal composed text:\n{user\_input}")

if \_\_name\_\_ == "\_\_main\_\_":

    main()

Output:



Program-02: Marketer to Machine: Develop level-2 Marketer-to-Machine (M2M) Scale of intelligent automation to personalize business email based on user preferences and interests (Extension to Exp. No.1).

Code:



import torch

from transformers import GPT2LMHeadModel, GPT2Tokenizer

import json

# ----------------------------------

# 1. Define User Profile & Intent

# ----------------------------------

user\_profile = {

    "name": "John Smith",

    "company": "TechNova Inc.",

    "interests": ["AI", "enterprise software", "cloud computing"],

    "email\_tone": "formal"

}

# ----------------------------------

# 2. Intent Selection Menu

# ----------------------------------

def choose\_intent():

    intents = {

        "1": "Cold outreach",

        "2": "Follow-up after meeting",

        "3": "Schedule a product demo",

        "4": "Thank you email",

        "5": "Partnership proposal"

    }

    print("Select Email Intent:")

    for k, v in intents.items():

        print(f"{k}. {v}")

    choice = input("Enter choice number: ").strip()

    return intents.get(choice, "Cold outreach")

# ----------------------------------

# 3. Email Prompt Builder

# ----------------------------------

def build\_prompt(profile, intent):

    prompt = f"""Write a {profile['email\_tone']} business email.

Sender: {profile['name']} from {profile['company']}.

Recipient is interested in: {', '.join(profile['interests'])}.

Goal: {intent}.

Email:

"""

    return prompt

# ----------------------------------

# 4. Load GPT-2 Medium

# ----------------------------------

def load\_model():

    print("Loading GPT-2 Medium model (1st time may take a minute)...")

    tokenizer = GPT2Tokenizer.from\_pretrained("gpt2-medium")

    model = GPT2LMHeadModel.from\_pretrained("gpt2-medium")

    return model, tokenizer

# ----------------------------------

# 5. Email Generator

# ----------------------------------

def generate\_email(prompt, model, tokenizer, max\_length=300):

    inputs = tokenizer.encode(prompt, return\_tensors="pt")

    outputs = model.generate(

        inputs,

        max\_length=max\_length,

        do\_sample=True,

        top\_k=40,

        top\_p=0.92,

        temperature=0.7,

        no\_repeat\_ngram\_size=2,

        pad\_token\_id=tokenizer.eos\_token\_id

    )

    result = tokenizer.decode(outputs[0], skip\_special\_tokens=True)

    generated = result[len(prompt):].strip()

    return generated

# ----------------------------------

# 6. Main CLI

# ----------------------------------

def main():

    intent = choose\_intent()

    prompt = build\_prompt(user\_profile, intent)

    model, tokenizer = load\_model()

    email = generate\_email(prompt, model, tokenizer)

    print("\n" + "-"\*50)

    print(f"📝 Personalized Email ({intent})\n")

    print(email)

    print("-"\*50)

if \_\_name\_\_ == "\_\_main\_\_":

    main()

Output:

Select Email Intent:

1. Cold outreach

2. Follow-up after meeting

3. Schedule a product demo

4. Thank you email

5. Partnership proposal

Enter choice number: 3

Personalized Email (Schedule a product demo)

Dear Sir/Madam,

I hope you're doing well. My name is John Smith, and I represent TechNova Inc., a leader in AI and enterprise software solutions. Based on your recent interest in cloud computing and smart automation, I'd love to schedule a quick demo of our latest product.

...

Program-03: AI and Marketing: Develop data-driven content for a given business organization (Web site). Optimize Website content for search engines. Send emails to customers with personalized content/activity.

Code:

from flask import Flask, request, render\_template\_string

from transformers import pipeline

import smtplib

from email.mime.text import MIMEText

app = Flask(\_\_name\_\_)

generator = pipeline("text-generation", model="gpt2")

HTML = """

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8" />

  <meta name="viewport" content="width=device-width, initial-scale=1.0" />

  <meta name="description" content="AI-powered marketing automation" />

  <title>AI Marketing Automation</title>

  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">

  <style>

    body { background-color: #f9f9f9; font-family: 'Segoe UI', sans-serif; }

    .hero { background: #343a40; color: white; padding: 60px 20px; text-align: center; }

    .section { padding: 40px 20px; }

    .footer { background: #212529; color: #aaa; padding: 20px; text-align: center; }

  </style>

</head>

<body>

  <div class="hero">

    <h1>AI-Powered Marketing Automation</h1>

    <p>Generate SEO content & personalized emails</p>

  </div>

  <div class="container section">

    <h2>🔍 SEO Content Generator</h2>

    <form method="post" action="/generate-content">

      <div class="mb-3">

        <label for="topic" class="form-label">Business Topic</label>

        <input type="text" name="topic" class="form-control" placeholder="e.g. Cloud CRM for SMEs" required>

      </div>

      <div class="mb-3">

        <label for="keywords" class="form-label">Target Keywords</label>

        <input type="text" name="keywords" class="form-control" placeholder="CRM, cloud, automation" required>

      </div>

      <button type="submit" class="btn btn-primary">Generate Content</button>

    </form>

    {% if content %}

      <div class="alert alert-secondary mt-3"><strong>Generated Content:</strong><br>{{ content }}</div>

    {% endif %}

  </div>

  <div class="container section">

    <h2>📬 Personalized Email Campaign</h2>

    <form method="post" action="/send-email">

      <div class="mb-3">

        <label for="name" class="form-label">Customer Name</label>

        <input type="text" name="name" class="form-control" required>

      </div>

      <div class="mb-3">

        <label for="email" class="form-label">Customer Email</label>

        <input type="email" name="email" class="form-control" required>

      </div>

      <div class="mb-3">

        <label for="activity" class="form-label">Recent Activity</label>

        <input type="text" name="activity" class="form-control" placeholder="e.g. Visited pricing page" required>

      </div>

      <button type="submit" class="btn btn-success">Send Email</button>

    </form>

    {% if message %}

      <div class="alert alert-info mt-3">{{ message }}</div>

    {% endif %}

  </div>

  <div class="footer">

    <p>&copy; 2025 Smart Marketing AI System</p>

  </div>

</body>

</html>

"""

def generate\_seo\_content(topic, keywords):

    prompt = f"Write an SEO-friendly paragraph about '{topic}' using these keywords: {', '.join(keywords)}"

    result = generator(prompt, max\_length=150, num\_return\_sequences=1)[0]['generated\_text']

    return result.strip()

def generate\_email(name, activity):

    prompt = f"Write a personalized marketing email to {name} who recently {activity}. Highlight our AI CRM benefits."

    result = generator(prompt, max\_length=150, num\_return\_sequences=1)[0]['generated\_text']

    return result.strip()

def send\_email(to\_email, subject, body):

    from\_email = "sreea8309@gmail.com"

    password = "wjwi gegh hwim cjtq"  # Use an app password, not your main password

    msg = MIMEText(body)

    msg['Subject'] = subject

    msg['From'] = from\_email

    msg['To'] = to\_email

    with smtplib.SMTP\_SSL('smtp.gmail.com', 465) as server:

        server.login(from\_email, password)

        server.send\_message(msg)

@app.route('/', methods=['GET'])

def home():

    return render\_template\_string(HTML)

@app.route('/generate-content', methods=['POST'])

def generate():

    topic = request.form['topic']

    keywords = [kw.strip() for kw in request.form['keywords'].split(',')]

    content = generate\_seo\_content(topic, keywords)

    return render\_template\_string(HTML, content=content)

@app.route('/send-email', methods=['POST'])

def email():

    name = request.form['name']

    email = request.form['email']

    activity = request.form['activity']

    content = generate\_email(name, activity)

    send\_email(email, "Thank You from Smart CRM", content)

    return render\_template\_string(HTML, message=f"Email sent to {name} ({email})")

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True)

Output:

Hi John,

Thanks for exploring our pricing page! We noticed your interest and would love to help you discover how our AI-powered CRM system can streamline your business processes, improve customer engagement, and save your valuable time.

Feel free to reach out if you have any questions or would like a personalized demo.

– The Smart CRM Team