Reg no: 212222040170

Exp 10: Content Creation (Reports, Articles, Case Studies, etc.) Using Prompt Patterns

Aim:

To demonstrate how various prompting techniques (query decomposition, decision-making, semantic filtering, etc.) can be employed to create content such as reports, articles, case studies, or creative works like comic books, using ChatGPT or similar models. The objective is to highlight how different prompt structures affect the content's quality, coherence, and structure.

PROCEDURE:

1. Select a Topic: Choose a topic (e.g., market trends, climate change, sci-fi story) that interests you.

2. Use Basic Prompts: Start by crafting simple, general prompts to generate initial content.

3. Refine Your Prompts: Gradually introduce more complexity by adding details, refining the structure, and using different prompt techniques like decision making or semantic filtering.

4. Evaluate the Output: Review the generated content for clarity, creativity, and accuracy.

5. Iterate for Improvement: Based on the feedback, refine the prompts and regenerate the content as needed.

TOOLS USED:

CHATGPT. GEMINI AI

OUTPUT:

USING AI TOOL FOR MINI PROJECT ON "ONLINE FAKE CURRENCY DETECTION USING MACHINE LEAERNING"

1. **First Draft: A basic draft of the report, case study, article, or story generated using simple prompts.**

Prompt: "Create a model for detecting fake currency vs original currency"

✅ Original Currency

- Download from Official Sources: Central bank websites often have high-resolution samples.
- Scan or photograph real notes using a high-resolution scanner or smartphone camera.

Confidential C

- Capture under different:
    - Lighting conditions
    - Angles and distances
    - Partial folds or wrinkles

❌ Fake Currency

- Use simulated fakes by printing or altering images digitally.
- Collect public domain examples (avoid any illegal practices).
- Use Photoshop or GIMP to alter features subtly (simulate fake notes).

⚠️ Legal Note: Never use counterfeit money. Use synthetic or publicly available fake data for educational/research purposes.

2. **Refined Content: A more detailed and structured output, achieved by applying more advanced prompt techniques.**

Prompt: “give a machine learning code for the model”

Model Code

python

CopyEdit

```
import os
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt

# Set parameters
IMG_SIZE = (128, 128)
BATCH_SIZE = 32
EPOCHS = 10

# Paths to training and testing folders
train_path = "currency_dataset/train"
test_path = "currency_dataset/test"

# Data augmentation for training data
```

Confidential C

```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    zoom_range=0.2,
    shear_range=0.2,
    horizontal_flip=True
)

# Only rescale for test data
test_datagen = ImageDataGenerator(rescale=1./255)

# Load train and test datasets
train_data = train_datagen.flow_from_directory(
    train_path,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='binary'
)

test_data = test_datagen.flow_from_directory(
    test_path,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='binary'
)

# CNN Model
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),

    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
```

Confidential C

```
    tf.keras.layers.Conv2D(128, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dropout(0.3),  # To prevent overfitting
    tf.keras.layers.Dense(1, activation='sigmoid')  # Binary classification
])

# Compile the model
model.compile(
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=['accuracy']
)

# Model summary
model.summary()

# Train the model
history = model.fit(
    train_data,
    epochs=EPOCHS,
    validation_data=test_data
)

# Save the model
model.save("currency_classifier.h5")

# Plot accuracy and loss
plt.plot(history.history['accuracy'], label='Train Accuracy')
```

Confidential C

```
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title("Model Accuracy")
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```

3. **Multiple Versions: Different versions of the content, showing the effect of prompt changes on the output.**

   Prompt: “Use machine learning model”

Fake Currency Detection using Traditional Machine Learning (SVM + HOG)

Python Code for the Classifier:

```
import os
import cv2
import numpy as np
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score
from skimage.feature import hog
import matplotlib.pyplot as plt

# Configuration
DATASET_DIR = 'currency_dataset'
IMAGE_SIZE = (128, 128)

def extract_hog_features(image):
    image = cv2.resize(image, IMAGE_SIZE)
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    features, _ = hog(gray,
                      orientations=9,
```

Confidential C

```
                pixels_per_cell=(8, 8),
                cells_per_block=(2, 2),
                block_norm='L2-Hys',
                visualize=True)
    return features

# Load dataset
X = []
y = []

for label, folder in enumerate(['real', 'fake']):
    folder_path = os.path.join(DATASET_DIR, folder)
    for filename in os.listdir(folder_path):
        filepath = os.path.join(folder_path, filename)
        img = cv2.imread(filepath)
        if img is not None:
            features = extract_hog_features(img)
            X.append(features)
            y.append(label)

X = np.array(X)
y = np.array(y)

# Split dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train SVM classifier
model = SVC(kernel='linear', probability=True)
model.fit(X_train, y_train)

# Predict and evaluate
y_pred = model.predict(X_test)
```

Confidential C

```
print("Accuracy:", accuracy_score(y_test, y_pred))

print("\nClassification Report:\n", classification_report(y_test, y_pred, target_names=['Real', 'Fake']))
```

Conclusion:

By applying various prompting techniques, you can generate high-quality content for a wide range of use cases, from business reports and case studies to creative works like short stories and articles. This experiment demonstrates how structured prompting can guide AI models like ChatGPT to create coherent, accurate, and engaging outputs tailored to specific needs.

Result:

The experiment successfully showed that using structured prompting techniques significantly improves the quality and relevance of AI-generated content. Whether applied to professional tasks such as business reports or creative projects like short stories, well-designed prompts helped guide the AI in producing coherent, context- aware, and purpose-driven outputs. The assistant responded effectively to different formats and objectives, confirming that prompt engineering is a key skill for maximizing the utility of language models like ChatGPT in diverse real-world applications.

Confidential C