import spacy

from transformers import GPT2LMHeadModel, GPT2Tokenizer

import torch

from textblob import TextBlob

# Load the spaCy model for traditional NLP tasks

nlp = spacy.load("en\_core\_web\_sm")

# Load the pre-trained GPT-2 model and tokenizer

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

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

def generate\_text(prompt, max\_length=100):

    """Generate text using GPT-2 model."""

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

    # Generate text using the GPT-2 model

    with torch.no\_grad():

        output = model.generate(input\_ids, max\_length=max\_length, num\_return\_sequences=1)

    # Decode the generated text

    generated\_text = tokenizer.decode(output[0], skip\_special\_tokens=True)

    return generated\_text

def analyze\_text(text):

    """Analyze text using spaCy for named entities and part-of-speech tagging."""

    doc = nlp(text)

    # Extract named entities

    entities = [(ent.text, ent.label\_) for ent in doc.ents]

    # Extract part-of-speech tags

    pos\_tags = [(token.text, token.pos\_) for token in doc]

    return entities, pos\_tags

def sentiment\_analysis(text):

    """Perform sentiment analysis using TextBlob."""

    blob = TextBlob(text)

    return blob.sentiment

def main():

    # Example prompt for text generation

    prompt = "In the future, artificial intelligence will"

    # Generate text

    generated\_text = generate\_text(prompt)

    print("Generated Text:")

    print(generated\_text)

    # Analyze the generated text

    print("\nAnalyzing Generated Text:")

    entities, pos\_tags = analyze\_text(generated\_text)

    print("Named Entities:")

    for entity in entities:

        print(f"{entity[0]} ({entity[1]})")

    print("\nPart-of-Speech Tags:")

    for pos in pos\_tags:

        print(f"{pos[0]} - {pos[1]}")

    # Perform sentiment analysis

    sentiment = sentiment\_analysis(generated\_text)

    print("\nSentiment Analysis:")

    print(f"Polarity: {sentiment.polarity}, Subjectivity: {sentiment.subjectivity}")

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

    main()

**Explanation of the Code**

1. **Text Generation**:
   * The **generate\_text** function takes a prompt and generates text using the GPT-2 model. It encodes the prompt, generates a sequence of text, and decodes it back to a human-readable format.
2. **Text Analysis**:
   * The **analyze\_text** function uses spaCy to analyze the generated text. It extracts named entities and part-of-speech tags, providing insights into the structure and meaning of the text.
3. **Sentiment Analysis**:
   * The **sentiment\_analysis** function uses TextBlob to perform sentiment analysis on the generated text, returning polarity and subjectivity scores.
4. **Main Execution**:
   * The script generates text based on a given prompt, analyzes the generated text for named entities and part-of-speech tags, and performs sentiment analysis.

**Usage**

* Run the script, and it will generate text based on the prompt, analyze the generated text for named entities and part-of-speech tags, and provide sentiment analysis results.

Python code integrates **traditional NLP** with **generative AI** using three key libraries:  
✅ **spaCy** – for Named Entity Recognition (NER) and Part-of-Speech (POS) tagging  
✅ **GPT-2** – for text generation using the transformers library  
✅ **TextBlob** – for sentiment analysis

**🔹 Breakdown of Functionality**

**1️Text Generation (generate\_text)**

* Uses a **GPT-2** model to generate text based on a given prompt.
* The generated text is **decoded** and returned.

**2️NLP Analysis (analyze\_text)**

* Uses **spaCy** to extract:
  + **Named Entities** (e.g., places, organizations, persons).
  + **Part-of-Speech (POS) Tags** (e.g., nouns, verbs, adjectives).

**3️Sentiment Analysis (sentiment\_analysis)**

* Uses **TextBlob** to measure:
  + **Polarity** (positive/negative sentiment).
  + **Subjectivity** (factual vs. opinion-based content).

**4️Main Execution (main)**

* Generates text using GPT-2.
* Analyzes the generated text for **NER & POS tags**.
* Performs **sentiment analysis**.

**🔹 Example Output**

plaintext

CopyEdit

Generated Text:

In the future, artificial intelligence will revolutionize the way we interact with technology.

Analyzing Generated Text:

Named Entities:

(None, since GPT-2 text might not contain named entities)

Part-of-Speech Tags:

In - ADP

the - DET

future - NOUN

, - PUNCT

artificial - ADJ

intelligence - NOUN

will - AUX

revolutionize - VERB

the - DET

way - NOUN

we - PRON

interact - VERB

with - ADP

technology - NOUN

. - PUNCT

Sentiment Analysis:

Polarity: 0.3, Subjectivity: 0.5

**🔹 Strengths**

✅ **Hybrid Approach** – Combines **traditional NLP (spaCy, TextBlob)** and **modern AI (GPT-2)**.  
✅ **Efficient Text Processing** – Covers **text generation, linguistic analysis, and sentiment evaluation**.  
✅ **Scalable** – Can be extended with more advanced models like **GPT-3, BERT, or LLaMA**.

**Key Points to Remember –**

**Linguistic Analysis: Understanding Language Structure and Meaning**

Linguistic analysis is the **systematic study of language** to understand its structure, meaning, and function. It helps in breaking down text or speech into components such as **syntax, semantics, phonetics, and pragmatics**.

In **Natural Language Processing (NLP)**, linguistic analysis is crucial for **text understanding, generation, and transformation**.

**🔹 Sentiment Analysis: Understanding Emotions in Text**

Sentiment Analysis, also known as **Opinion Mining**, is a **Natural Language Processing (NLP)** technique used to **analyze and determine the sentiment (emotion or opinion) expressed in a piece of text**.

It helps in identifying whether the text conveys **positive, negative, or neutral** emotions.

**🔹 How Sentiment Analysis Works**

**1️Text Preprocessing**

* **Tokenization** – Splitting text into words or phrases.
* **Removing Stopwords** – Filtering out common words like *"is", "the", "and"*.
* **Stemming/Lemmatization** – Converting words to their root form (*"running" → "run"*).

**2️Feature Extraction**

* **Bag-of-Words (BoW)** – Counts word occurrences.
* **TF-IDF (Term Frequency - Inverse Document Frequency)** – Measures word importance.
* **Word Embeddings (Word2Vec, GloVe, BERT)** – Captures word meaning in vector form.

**3️Sentiment Classification**

* **Rule-Based Approach** – Uses predefined dictionaries of positive/negative words.
* **Machine Learning Models** – Uses algorithms like SVM, Naïve Bayes, Decision Trees.
* **Deep Learning Models** – Uses LSTMs, Transformers (BERT, GPT) for better accuracy.

**🔹 Sentiment Polarity Levels**

1️ **Positive Sentiment** → *"I love this product!"* 😊  
2️ **Negative Sentiment** → *"This service is terrible."* 😡  
3️ **Neutral Sentiment** → *"The weather is okay today."* 😐

Some advanced models also detect:  
🔹**Mixed Sentiment** → *"The movie had great visuals but a poor storyline."*  
🔹**Emotion Detection** → Happy, Sad, Angry, Excited, etc.