Week 3 Lab: LLM APIs & Multimodal AI

CS 203: Software Tools and Techniques for AI

Duration: 3 hours

Lab Overview

By the end of this lab, you will:

- Set up and use Gemini API
- Perform text classification and NER
- Analyze images with vision models
- Extract data from PDFs
- Build multimodal AI applications

Structure:

- Part 1: Setup & Text Tasks (45 min)
- Part 2: Vision Tasks (60 min)
- Part 3: Document & Audio (60 min)
- Part 4: Build Your App (15 min)

Setup (15 minutes)

Get API Key

1. Visit https://aistudio.google.com/apikey
2. Create API key
3. Set environment variable:

```
export GEMINI_API_KEY='your-key-here'
```

Install Packages

```
pip install google-genai pillow requests matplotlib pandas numpy
```

Verify Setup

Exercise 1.1: Sentiment Analysis (15 min)

Create a sentiment classifier for product reviews.

```
MODEL = "models/gemini-3-pro-preview"

reviews = [
    "This product exceeded expectations!",
    "Terrible quality, broke after one day.",
    "It's okay, nothing special."
]

for review in reviews:
    response = client.models.generate_content(
        model=MODEL,
        contents=f"Sentiment (Positive/Negative/Neutral): {review}"
    )
    print(f"{review[:30]}... => {response.text}")
```

Task: Add confidence scores and batch processing.

Exercise 1.2: Named Entity Recognition (15 min)

Extract entities from news articles.

```python
text = "Apple CEO Tim Cook met PM Modi in Delhi on Monday."

prompt = f"""
Extract entities as JSON:
{{"Person": [], "Organization": [], "Location": [], "Date": []}}

Text: {text}
"""


response = client.models.generate_content(model=MODEL, contents=prompt)
import json
entities = json.loads(response.text)
print(entities)
```

Task: Process multiple articles, create entity frequency analysis.

Exercise 2.1: Image Description (20 min)

Describe and tag images automatically.

```python
from PIL import Image
IMAGE_MODEL = "models/gemini-3-pro-image-preview"

image = Image.open("product.jpg")

response = client.models.generate_content(
    model=IMAGE_MODEL,
    contents=["Describe this image and suggest 5 tags.", image]
)

print(response.text)
```

Task: Batch process images, extract structured tags as JSON.

Exercise 2.2: OCR and Data Extraction (20 min)

Extract text from documents.

```
doc_image = Image.open("receipt.jpg")

prompt = """
Extract receipt data as JSON:
{
  "merchant": "",
  "date": "",
  "items": [{"name": "", "price": 0}],
  "total": 0
}
"""

response = client.models.generate_content(
    model=IMAGE_MODEL,
    contents=[prompt, doc_image]
)

receipt = json.loads(response.text)
print(f"Total: ${receipt['total']}")
```

Exercise 2.3: Object Detection (20 min)

Detect objects with bounding boxes.

```python
image = Image.open("street.jpg")

prompt = """
Detect objects. Return JSON array:
[{"object": "car", "bbox": [x1,y1,x2,y2], "confidence": 0.95}]
Normalize coordinates to 0-1000.
"""


response = client.models.generate_content(
    model=IMAGE_MODEL,
    contents=[prompt, image]
)

detections = json.loads(response.text)
# Draw boxes (code provided in solutions)
```

Exercise 3.1: PDF Data Extraction (20 min)

Extract structured data from PDFs.

```python
pdf_file = client.files.upload(path="invoice.pdf")

prompt = """
Extract invoice data as JSON:
{
  "invoice_number": "",
  "date": "",
  "vendor": "",
  "line_items": [],
  "total": 0
}
"""

response = client.models.generate_content(
    model=MODEL,
    contents=[prompt, pdf_file]
)

data = json.loads(response.text)
```

Exercise 3.2: Audio Transcription (20 min)

Transcribe audio files.

```python
audio_file = client.files.upload(path="interview.mp3")

# Wait for processing
import time
while audio_file.state == "PROCESSING":
    time.sleep(2)
    audio_file = client.files.get(audio_file.name)

response = client.models.generate_content(
    model=MODEL,
    contents=["Transcribe with speaker labels.", audio_file]
)

print(response.text)
```

Exercise 3.3: Video Summarization (20 min)

Analyze video content.

```
video_file = client.files.upload(path="demo.mp4")

while video_file.state == "PROCESSING":
    time.sleep(5)
    video_file = client.files.get(video_file.name)

response = client.models.generate_content(
    model=MODEL,
    contents=[
        "Summarize this video in 3 bullet points.",
        video_file
    ]
)

print(response.text)
```

Part 4: Build Your Application (15 min)

Choose one:

1. **Product Review Analyzer**
   - Sentiment + entity extraction
   - Generate summary report
2. **Document Processor**
   - Upload multiple PDFs
   - Extract and aggregate data
3. **Image Cataloging System**
   - Auto-tag images
   - Generate descriptions

Deliverables

1. Python notebook with all exercises
2. At least one complete application
3. Brief report (README.md):
    - What you built
    - Challenges faced
    - API costs incurred
    - Potential improvements

Bonus: Deploy as web app with Streamlit or Gradio

Excellent Work!

Next week: Advanced topics and production deployment

Questions? Office hours tomorrow 3-5 PM