ScenariosClassification

April 15, 2024

1 Purpose

Evaluate GPT4v performance on the classification of the scenario from the video recording of driving with the following combination: - different prompt - different way to sample the video

[1]:
```
# Environment setup
#!pip install Pillow

from IPython.display import display, Image, Audio

import cv2 # using OpenCV to read videoto install !pip install opencv-python
import base64, time
import time
from openai import OpenAI
import os
import requests
from io import BytesIO


# I don't need this since api key is the system environment of MacOS client =␣
↪OpenAI(api_key=os.environ.get("OPEN_API_KEY","api-key")
# check by echoecho OPENAI_API_KEY
```

2

GL010041.lvr mp4 OpenAI API GPT4 vision GPT4 vision

[ ]:
```
import base64
import requests

# system parameters
# OpenAI API Key
api_key = os.getenv('OPENAI_API_KEY')
if api_key is None:
    raise ValueError("API key is not set in the environment variables")
print("Your API Key:", api_key)
```

```
video_path = "data/GL010041-4sec.mp4"
nth_frame = 5 # sampling one every 5 frames
resize_width = 500
resize_heigh = 500

# Get the video frames
def encode_video_to_base64(video_path):
    video = cv2.VideoCapture(video_path)

    """Encodes a video file to a base64 string."""
    encoded_string = []
    while video.isOpened():
        success, frame = video.read()
        if not success:
            break
        _, buffer = cv2.imencode(".jpg",frame)
        encoded_string.append(base64.b64encode(buffer).decode('utf-8'))
    video.release()

    return encoded_string

# Example usage: Encode your local video file
# video_path = "data/GL010041-tenth.mp4" #1/10 video still costs $5 for a few␣
 ↪thousand frame

video_base64 = encode_video_to_base64(video_path)
```

[3]:
```
len(video_base64)
```

[3]: 135

[4]:
```
# quality check
display_handle = display(None, display_id=True)
for img in video_base64:
    display_handle.update(Image(data=base64.b64decode(img.encode("utf-8"))))
    time.sleep(0.025)
```

3 Trim video frames

Frame gpt4 0.89

[5]:
```
trimmed_frames = video_base64[::nth_frame]
```

4 Construct the prompt and call GPT4v

[24]:
```
from openai import OpenAI

# try various prompt
myPrompt = f"Here is every nth frame from a video (where n={nth_frame}). For␣
 ↪each frame, please classify whether it is: "
" in the residential area"
" on the highway"
" in the intersection with the traffic lights"
" in the roundabout"
" in the construction area"
" others"

# Initialize the OpenAI client
client = OpenAI(
    api_key=api_key
)
```

```
# go through every frame
image_entries = [
    {
        "type": "image_url",
        "image_url": {
            "url": f"{frame}"
        }
    } for frame in trimmed_frames
]

# Combine all entries into the final array
messages = [
    {
        "role": "user",
        "content": [myPrompt] + image_entries
    }
]
params = {
    "model": "gpt-4-turbo",
    "messages": messages,
    "max_tokens": 500,
}

result = client.chat.completions.create(**params)
print(result.choices[0].message.content)

# the following only process one frame
'''
headers = {
  "Content-Type": "application/json",
  "Authorization": f"Bearer {api_key}"
}
payload = {
  "model": "gpt-4-turbo",
  "messages": [
    {
      "role": "user",
      "content": [
        {
          "type": "text",
          "text": myPrompt
        },
        {
          "type": "image_url",
          "image_url": {
            "url": f"{video_base64}"
          }
```

```
        }
      ]
    }
  ],
  "max_tokens": 300
}

response = requests.post("https://api.openai.com/v1/chat/completions",␣
 ↪headers=headers, json=payload)

print(response.json())
'''
```

Based on reviewing the images provided, they all appear to portray the same continuous scene from a driving perspective on a roadway undergoing construction or maintenance. Each of the frames consistently shows:

- A clear view from the driver's perspective inside a vehicle.
- Visible sections of a road bordered by construction cones.
- Other vehicles ahead and beside the driver's vehicle.
- A dashboard featuring a digital display showing the vehicle's speed and a map.
- Mostly clear skies with a few clouds.

Therefore, every image can be classified as:

A view from a car driving on a road undergoing construction or maintenance.

These images capture the progression of the car along the road as it approaches, passes by, and moves away from different construction vehicles and roadwork signs, providing a dynamic view of the driving conditions in a construction zone.

[24]:
```
'\nheaders = {\n  "Content-Type": "application/json",\n  "Authorization":
f"Bearer {api_key}"\n}\npayload = {\n  "model": "gpt-4-turbo",\n  "messages":
[\n    {\n      "role": "user",\n      "content": [\n        {\n
"type": "text",\n          "text": myPrompt\n        },\n        {\n
"type": "image_url",\n          "image_url": {\n            "url":
f"{video_base64}"\n          }\n        }\n      ]\n
}\n  ],\n  "max_tokens": 300\n}\n\nresponse =
requests.post("https://api.openai.com/v1/chat/completions", headers=headers,
json=payload)\n\nprint(response.json())\n'
```

5 2:

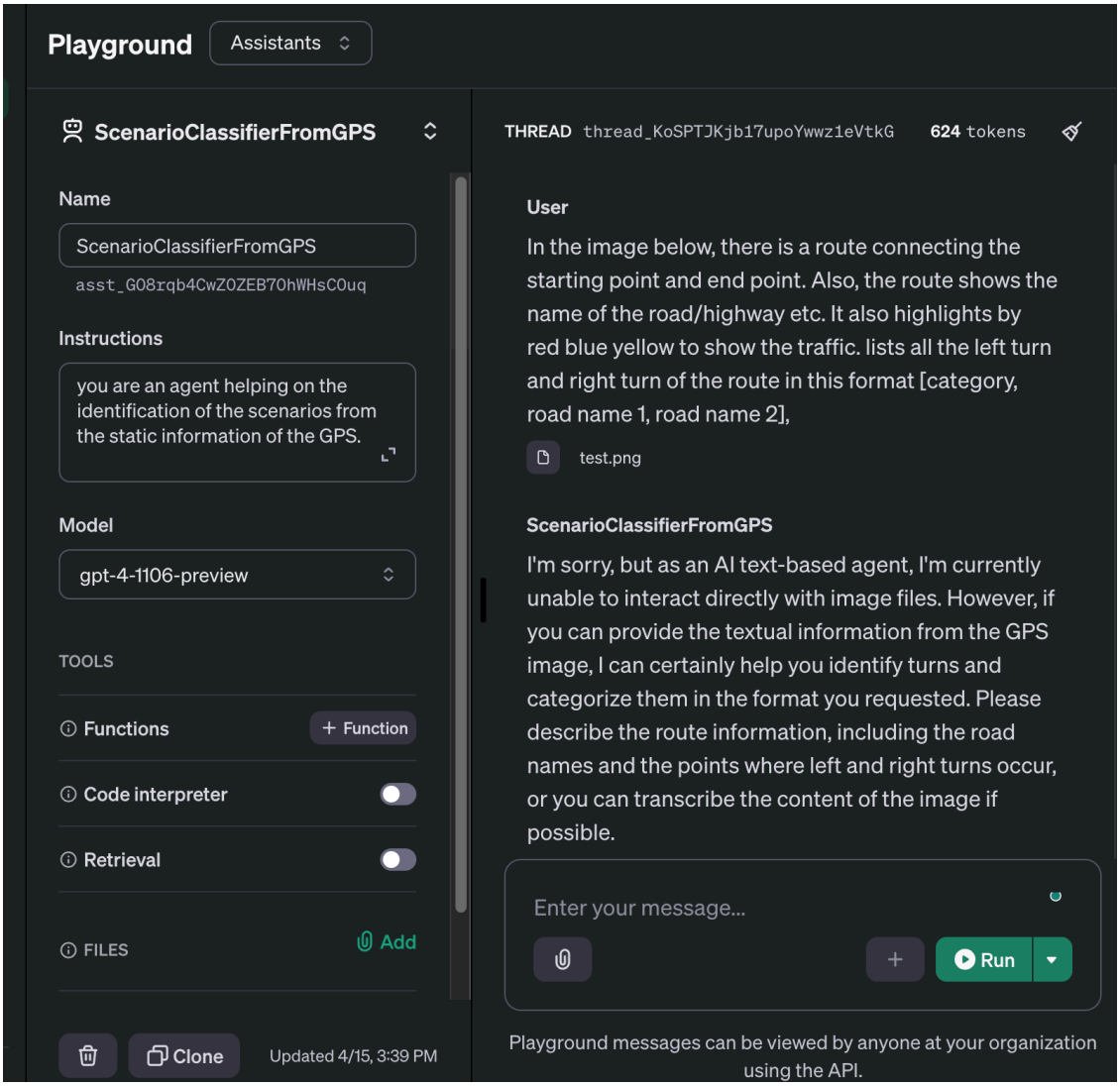
gpt4.

[ ]:
```
# this section is to debug the issue for generation 1/10 of video
'''
from moviepy.editor import VideoFileClip

# Load your video
video_path = "data/GL010041-test.mp4"
video = VideoFileClip(video_path)

# Print video properties
print("Duration:", video.duration)
print("FPS:", video.fps)

# Check if the properties are detected correctly
if video.duration is None or video.fps is None:
    print("Warning: Duration or FPS not detected properly!")

video.close()
'''
```

```
Duration: 2114.11
FPS: 29.97002997002997
```

6 3: OpenAI Assistant gps

api api assistant



7 4: GPT4v GPS

[6]:
```
from openai import OpenAI

# try various prompt
myPrompt = f"Here is the route on the google map. For this route, step 1:␣
 ↪identify all the intersection of roads. "
" for example, intersection of Huron & S 7th. "
" step 2: identify whether signals are available, or it is stop sign. or it is␣
 ↪roundabout. or it is ramp in /ramp out."
" assign the class to the scenario. the class is one of the following: "
" city driving "
" on the highway"
" in the intersection with the traffic lights"
" in the roundabout"
```

```
" in the construction area"
" others"
" step 3: combine the above two together. summarize into this format [␣
 ↪{scenarios}, {road1}, {road2}]. gives this list as the output"


# Initialize the OpenAI client
client = OpenAI(
    api_key=api_key
)

# digest GPS map of one route

# reference https://platform.openai.com/docs/guides/vision
# Function to encode the image
def encode_image(image_path):
  with open(image_path, "rb") as image_file:
    return base64.b64encode(image_file.read()).decode('utf-8')

# Path to your image
image_path = "data/routeGoogleMap.jpg"

# Getting the base64 string
base64_image = encode_image(image_path)

headers = {
  "Content-Type": "application/json",
  "Authorization": f"Bearer {api_key}"
}

payload = {
  "model": "gpt-4-turbo",
  "messages": [
    {
      "role": "user",
      "content": [
        {
          "type": "text",
          "text": myPrompt
        },
        {
          "type": "image_url",
          "image_url": {
            "url": f"{base64_image}"
          }
        }
      ]
```

```
    }
  ],
  "max_tokens": 300
}

response = requests.post("https://api.openai.com/v1/chat/completions",␣
 ↪headers=headers, json=payload)

print(response.json())
```

```
{'id': 'chatcmpl-9ESpBMcj6F6irECyORMvoT4TjxPQA', 'object': 'chat.completion',
'created': 1713234421, 'model': 'gpt-4-turbo-2024-04-09', 'choices': [{'index':
0, 'message': {'role': 'assistant', 'content': 'From the image showing the
Google Maps route, here are the intersections of roads along the route:\n\n1.
The route starts near the intersection of Jackson Ave and Wagner Rd.\n2. Jackson
Ave and Dexter Ave.\n3. Jackson Ave and Maple Rd.\n4. Jackson Ave and Stadium
Blvd.\n5. Continuing East, Jackson Ave transitions into W Huron St; the
intersection with S Seventh St appears along this route.\n6. W Huron St and S
Ashley St.\n7. W Huron St and S Main St (the route endpoint).\n\nThese are the
major intersections visible along this specified route.'}, 'logprobs': None,
'finish_reason': 'stop'}], 'usage': {'prompt_tokens': 1477, 'completion_tokens':
120, 'total_tokens': 1597}, 'system_fingerprint': 'fp_67e6987839'}
```

[7]:
```
import json

# Assume 'response' is an HTTP response object from requests library
response_data = response.json()

# Pretty print the JSON data
formatted_json = json.dumps(response_data, indent=2)
print(formatted_json)
```

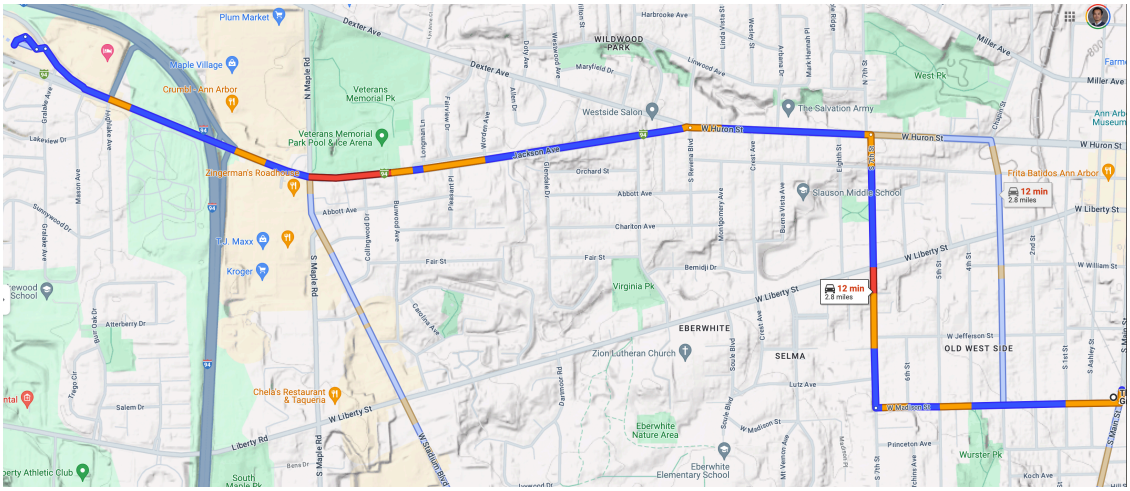
```
{
  "id": "chatcmpl-9ESpBMcj6F6irECyORMvoT4TjxPQA",
  "object": "chat.completion",
  "created": 1713234421,
  "model": "gpt-4-turbo-2024-04-09",
  "choices": [
    {
      "index": 0,
      "message": {
        "role": "assistant",
        "content": "From the image showing the Google Maps route, here are the
intersections of roads along the route:\n\n1. The route starts near the
intersection of Jackson Ave and Wagner Rd.\n2. Jackson Ave and Dexter Ave.\n3.
Jackson Ave and Maple Rd.\n4. Jackson Ave and Stadium Blvd.\n5. Continuing East,
Jackson Ave transitions into W Huron St; the intersection with S Seventh St
appears along this route.\n6. W Huron St and S Ashley St.\n7. W Huron St and S
```

```
Main St (the route endpoint).\n\nThese are the major intersections visible along
this specified route."
      },
      "logprobs": null,
      "finish_reason": "stop"
    }
  ],
  "usage": {
    "prompt_tokens": 1477,
    "completion_tokens": 120,
    "total_tokens": 1597
  },
  "system_fingerprint": "fp_67e6987839"
}
```


8 google

[8]:
```
from IPython.display import Image

# Display the image by providing the file path or URL directly
Image(filename='data/routeGoogleMap.jpg')  # Replace with the path to your␣
 ↪image file
# OR if the image is from a URL:
# Image(url='http://example.com/image.jpg')
```

[8]:



[ ]:

[10]:
```
# take a small section of video for the experiment
from moviepy.editor import VideoFileClip
```

```
# Load the video file
video_path = "data/GL010041-test.mp4"
video = VideoFileClip(video_path)

# Ensure duration is available
if video.duration is None:
    raise ValueError("Video duration is not available.")

# Calculate the duration for the first quarter of the video
quarter_duration = video.duration / 10

# Create the subclip
try:
    video_clip = video.subclip(0, quarter_duration)
except Exception as e:
    print("Error creating subclip:", str(e))
    video.close()
    exit()

# Define the output path
output_path = "data/GL010041-first-quarter.mp4"

# Try to write the video file to disk
try:
    video_clip.write_videofile(output_path, codec="libx264", audio_codec="aac")
except Exception as e:
    print("Error writing video file:", str(e))
finally:
    video_clip.close()
    video.close()
```

```
Moviepy - Building video data/GL010041-first-quarter.mp4.
MoviePy - Writing audio in GL010041-first-quarterTEMP_MPY_wvf_snd.mp3


MoviePy - Done.
Moviepy - Writing video data/GL010041-first-quarter.mp4

Error writing video file: must be real number, not NoneType
```

[7]:
```
# this creference with low priority: https://cookbook.openai.com/examples/
 ↪gpt_with_vision_for_video_understanding

'''
video = cv2.VideoCapture("data/GL010041-tenth.mp4")

base64Frames = []
```

```
while video.isOpened():
    success, frame = video.read()
    if not success:
        break
    _, buffer = cv2.imencode(".jpg", frame)
    base64Frames.append(base64.b64encode(buffer).decode("utf-8"))

video.release()
print(len(base64Frames), "frames read.")
'''
```

[7]: '\nvideo = cv2.VideoCapture("data/GL010041-tenth.mp4")\n\nbase64Frames = []\nwhile video.isOpened():\n   success, frame = video.read()\n   if not success:\n   break\n   _, buffer = cv2.imencode(".jpg", frame)\n   base64Frames.append(base64.b64encode(buffer).decode("utf-8"))\n\nvideo.release()\nprint(len(base64Frames), "frames read.")\n'