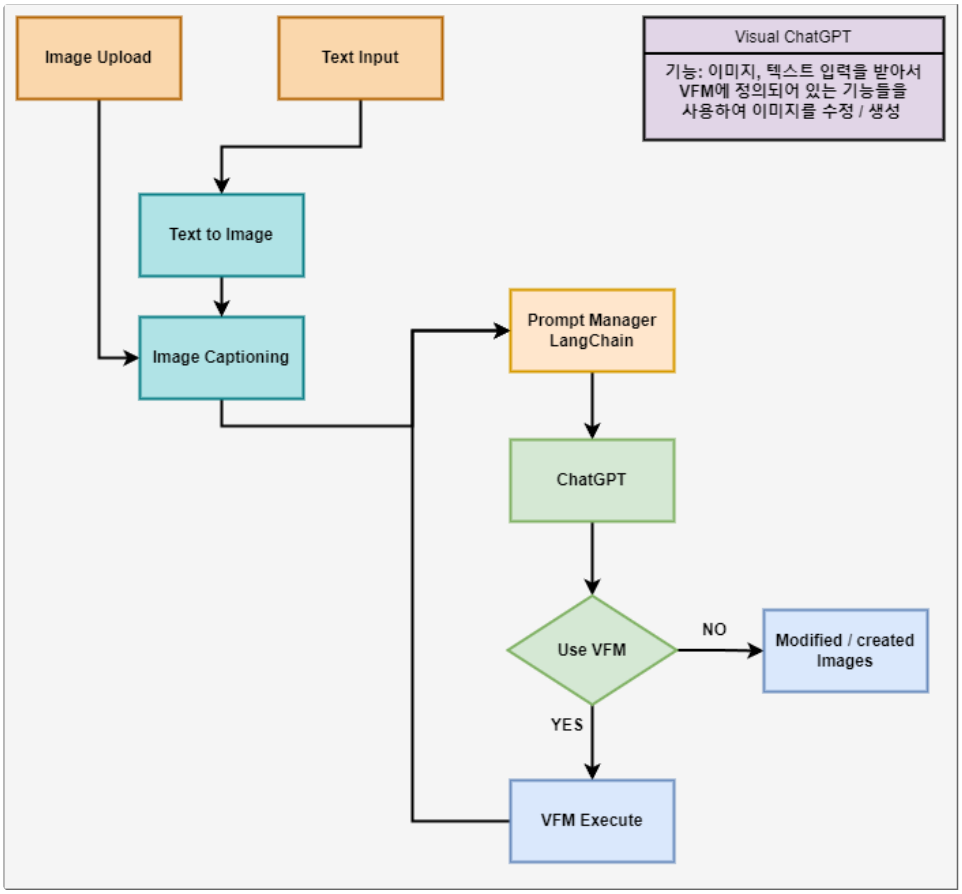
Visual ChatGPT API Documents

Visual ChatGPT API Documents Table of contents

Original Code

★ API 기능 요약



이미지, 텍스트를 입력 받아 VFM의 기능들을 이용하여 Text-to-Image, Image Captioning을 진행 후, 수정하고자 하는 부분을 입력하여 이미지를 수정, 생성하는 역할을 하는 API

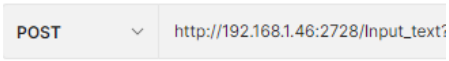
★ API 사용 설명서

> **해당 API는 2가지의 방법으로 불러야 함.**
>
> **공통적으로 2728번의 포트 번호를 가지고 시행.**

✔ Input_text

Methods: POST

end_point: /Input_text



parameters: text(text: value 형식)

KEY	VALUE
text	이 이미지 위에 고양이 그려줘

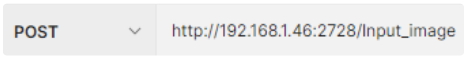
Response: 해당 이미지.png

기능: 이미지를 초기에 업로드할 때만 빼고 대부분 Input_text를 사용. 해당 end_point를 호출해야 VFM를 통해 나온 수정/생성된 이미지를 확인할 수 있음.

✔ Input_image

Methods: POST

end_point: /Input_Image



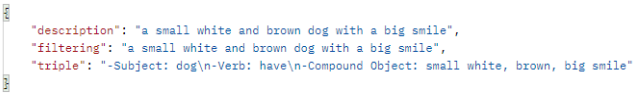
parameters: form-data (image/file,

NER(사용할 시에만) – True(True로 요청해야 필터링, 트리플 추출))

KEY	VALUE
text	이 이미지 위에 고양이 그려줘

Response: 해당 이미지.png

NER(True) - description, Triple



기능: 이미지를 초기에 업로드할 때만 사용. 또한 중간에 다른 이미지를 업로들 할 때 사용.

★ API 코드 설명

- **Visual ChatGPT Prefix**

> **ChatGPT에 Visual ChatGPT가 어떠한 기능을 하는 모델인지 설명하는 Prompt**

```
VISUAL_CHATGPT_PREFIX = """Visual ChatGPT is designed to be able to assist with a wide range of text and visual r
Visual ChatGPT is able to process and understand large amounts of text and images. As a language model, Visual Ch
Human may provide new figures to Visual ChatGPT with a description. The description helps Visual ChatGPT to under
Overall, Visual ChatGPT is a powerful visual dialogue assistant tool that can help with a wide range of tasks and
TOOLS:
------
Visual ChatGPT  has access to the following tools:"""
```

- **Visual ChatGPT Format Instructions**

> **ChatGPT에 Prompt를 보낼 때, 해당하는 Tool(VFM)을 사용할지 여부 Prompt**

```
VISUAL_CHATGPT_FORMAT_INSTRUCTIONS = """To use a tool, please use the following format:
```
Thought: Do I need to use a tool? Yes
Action: the action to take, should be one of [{tool_names}]
Action Input: the input to the action
Observation: the result of the action
```
When you have a response to say to the Human, or if you do not need to use a tool, you MUST use the format:
```
Thought: Do I need to use a tool? No
{ai_prefix}: [your response here]
```
"""
```

- **Visual ChatGPT Suffix**

여태까지 진행한 Chat History와 새로운 요구 사항을 시행하기 위한 Prompt

```
VISUAL_CHATGPT_SUFFIX = """You are very strict to the filename correctness and will never fake a file name if it
You will remember to provide the image file name loyally if it's provided in the last tool observation.
Begin!
Previous conversation history:
{chat_history}
New input: {input}
Since Visual ChatGPT is a text language model, Visual ChatGPT must use tools to observe images rather than imagi
The thoughts and observations are only visible for Visual ChatGPT, Visual ChatGPT should remember to repeat impo
Thought: Do I need to use a tool? {agent_scratchpad} Let's think step by step.
"""
```

- **VFM Model**

VFM의 기능들을 사용하기 위한 VFM모델 정의

```
class Text2Image: # Text to Image를 하기 위한 VFM 모델 정의 부분
    def __init__(self, device):
        print(f"Initializing Text2Image to {device}")
        self.device = device
        self.torch_dtype = torch.float16 if 'cuda' in device else torch.float32
        self.pipe = StableDiffusionPipeline.from_pretrained("runwayml/stable-diffusion-v1-5",
                                                            torch_dtype=self.torch_dtype)
        self.pipe.to(device)
        self.a_prompt = 'best quality, extremely detailed'
        self.n_prompt = 'longbody, lowres, bad anatomy, bad hands, missing fingers, extra digit, ' \
                        'fewer digits, cropped, worst quality, low quality'

    # 해당 Prompt를 Prompt Manager으로 보냄으로써, 해당 기능이 수행되도록 하는 Prompt
    @prompts(name="Generate Image From User Input Text",
             description="useful when you want to generate an image from a user input text and save it to a file
                         "like: generate an image of an object or something, or generate an image that includes
                         "The input to this tool should be a string, representing the text used to generate imag
    def inference(self, text):
        image_filename = os.path.join('image', f"{str(uuid.uuid4())[:8]}.png") # 해당 이미지 가져오기
        prompt = text + ', ' + self.a_prompt
        image = self.pipe(prompt, negative_prompt=self.n_prompt).images[0]
```

```
        image.save(image_filename) # 이미지 저장
        print(
            f"\nProcessed Text2Image, Input Text: {text}, Output Image: {image_filename}")
        return image_filename
```

- **Load VFM Model**

> **VFM 기능을 사용하기 위해서는 각 Class별로 정의된 기능들을 불러와야 함.**

```
load_dict = {'ImageCaptioning': 'cuda:0'} # 이부분에 'Image2Canny': 'cpu'와 같이 추가하여 모델 로드
```

- **Pre-processing with ChatGPT**

> **전처리를 위한 ChatGPT를 사용하기 위한 기능**

```
def translateGPT(text, model="text-davinci-003", max_tokens=1000, top_p=1.0): # model: davinci-003 사용
    response = openai.Completion.create(
      model=model,
      prompt=text, # 원하고자 하는 Prompt 작성
      temperature=0.5, # 생성된 텍스트의 다양성을 조절
      max_tokens=max_tokens, # 텍스트 최대 길이 조절
      top_p=top_p, # 토큰 샘플링에 대한 확률 조절
      frequency_penalty=0.0, # 생성된 텍스트에서 자주 발생하는 토큰에 패널티 부여 조절
      presence_penalty=0.0 # 입력에 이미 사용된 토큰테 페널티 부여 조절
    )
    return response.to_dict()['choices'][0]["text"].lstrip("\n")
```

- **Text Input**

> **이미지를 업로드하지 않고 텍스트를 통해 이미지를 생성하고 싶을 때, 사용**
>
> **VFM 기능을 이용해 처리된 이미지를 처리할 때, 사용**

```
@app.route('/Input_text', methods=['POST']) # Input_text로 호출하여 사용
def run_text():
    text = request.args.get('text', 'cat') # 사용자가 입력한 text로 생성(default로 cat생성)

    text = f"Translate korean to english: " + f"{text}" # 한글 -> 영어
    text = translateGPT(text) # ChatGPT를 통해 전처리

    state = session.get('state', []) # session 설정(대화 기록을 저장하기 위함)
    # text: 입력 문장 / state: 대화 기록 저장
    agent.memory.buffer = cut_dialogue_history(agent.memory.buffer, keep_last_n_words=500) # 대화 내용 기억

    res = agent({"input": text.strip()})
    res['output'] = res['output'].replace("\\", "/")
    response = re.sub('(image/[-\w]*.png)', lambda m: f'![](/file={m.group(0)})*{m.group(0)}*', res['output'])

    # 이미지 찾기
    if not os.path.exists(dir_path):
        os.makedirs(dir_path)
        os.chmod(dir_path, stat.S_IWUSR)

```

```
    # canny edge
    if re.search(r'(\w+_edge_\w+\.png)', response):
        Image_name = re.search(r'(\w+_edge_\w+\.png)', response).group(1)
        send_file_dir = os.path.join(dir_path, Image_name)
        res = send_file(send_file_dir, as_attachment=True)

    # text2image
    elif re.search(r'image/([^\s]+)\.png', response):
        Image_name = re.search(r'image/([^\s]+)\.png', response).group(1)
        Image_name = re.search(r'([^\s]+)\.png', Image_name).group(1) + '.png'
        send_file_dir = os.path.join(dir_path, Image_name)
        res = send_file(send_file_dir, as_attachment=True)

    # replace image object
    elif re.search(r'(\w+_replace-something_\w+\.png)', response):
        Image_name = re.search(r'(\w+_replace-something_\w+\.png)', response).group(1)
        Image_name = re.search(r'([^\s]+)\.png', Image_name).group(1) + '.png'
        send_file_dir = os.path.join(dir_path, Image_name)
        res = send_file(send_file_dir, as_attachment=True)

    # painting (pix2pix)
    elif re.search(r'(\w+_pix2pix_\w+\.png)', response):
        Image_name = re.search(r'(\w+_pix2pix_\w+\.png)', response).group(1)
        Image_name = re.search(r'([^\s]+)\.png', Image_name).group(1) + '.png'
        send_file_dir = os.path.join(dir_path, Image_name)
        res = send_file(send_file_dir, as_attachment=True)

    else:
        res = response

    # state 리스트에 새로운 튜플을 추가합니다.
    state.append((text, response))

    # 새로운 state 리스트를 세션에 저장합니다.
    session['state'] = state
    print(f"\nProcessed run_text, Input text: {text}\nCurrent state: {state}\n"
          f"Current Memory: {agent.memory.buffer}")

    return res
```

- **Image Input**

> **이미지를 업로드할 때, 사용**
>
> **parameter에 NER을 추가하면 해당 이미지에 대한 desctiption, triple 추출**

```
@app.route('/Input_image', methods=['POST']) # Input_image를 호출하여 사용
def run_image():
    try:
        state = session['state'] # 대화 내용 기억하기 위함
    except:
        state = session.get('state', [])

    # NER Check
    ner = request.form.get('NER', False) # NER Parameter 존재 여부 확인

    # Check pass (이미지를 저장할 곳 확인)
```

```
    if not os.path.exists(dir_path):
        os.makedirs(dir_path)
        os.chmod(dir_path, stat.S_IWUSR)

    # Save Image, But Change the extension to .png
    image = request.files['image']
    image_filename = f"{str(uuid.uuid4())[:8]}.png" # uuid를 부여함으로써 새로운 이름 부여
    with tempfile.TemporaryDirectory() as temp_dir:
        temp_image_path = os.path.join(temp_dir, 'temp_image.png')
        image.save(temp_image_path)
        shutil.move(temp_image_path, os.path.join(dir_path, image_filename))

    # Make description and prompt
    description = models['ImageCaptioning'].inference(os.path.join(dir_path, image_filename))
    Human_prompt = f'\nHuman: provide a figure named {image_filename}. The description is: {description}. This i
    AI_prompt = "Received.  "
    agent.memory.buffer = agent.memory.buffer + Human_prompt + 'AI: ' + AI_prompt
    state = state + [(f"![](/file={image_filename})*{image_filename}*", AI_prompt)]
    session['state'] = state
    print(f"\nProcessed run_image, Input image: {image_filename}\nCurrent state: {state}\n"
          f"Current Memory: {agent.memory.buffer}")

    if ner: # NER Parameter가 존재하면 description, triple, filtering 등 반환
        res = {}
        res['description'] = description

        text = f"please extract the nouns from this sentence: " + f"{description}"
        noun_list = translateGPT(text)
        pattern = re.compile("[^ \na-zA-Z]+")
        noun_list = pattern.sub("", noun_list)
        noun_list = re.sub('[ \n\t]', ',', noun_list).replace(',,', ',')
        mood = 'bright'
        # res['noun_list'] = noun_list

        res['filtering'] = translateGPT(f"""
            Filter out sexually explicit, political, and violent content from the sentence '{description}' and r
            If there are no corresponding words, return the original sentence. Slang terms will not be considere
            """).strip()

        # res['replace'] = translateGPT(f"""
        #     After understanding the intent of the sentence and Words that violate openai policy, such as sensa
        #     If a substitute phrase is violated of openai policy, a warning phrase is printed.
        #     """).strip()

        # if not re.findall(r'and \*{2,}|\*{2,}', res['filtering']):
        if res['filtering'].lower() == description:
            res['triple'] = translateGPT(
                    f"""
                    Extract RDF triple from the {description} and separate the subject, verb, and object but, Al
                    Always return the output in JSON format as '-Subject: , -Verb: , -Compound Object: '.
                    """).strip()

        # res['test'] = translateGPT(
        #     f"""
        #     Please create a natural sentence by adding {mood} atmosphere to {description}.
        #     """
        # )
        return res
```

```
70      else: # NER Parameter가 존재하지 않으면 이미지 반환
71          return send_file(os.path.join(dir_path, image_filename), as_attachment=True)
```

- **Run Flask API**

> **Flask API를 실행시키기 위한 부분**
>
> **해당 Port Number는 2728로 설정됨.**

```
# host='0.0.0.0'으로 설정함으로써 모든 접근 허용 처리
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=2728, threaded=True, debug=True, use_reloader=False)
```