오픈소스 SW 기여

프로젝트 설계서

박찬호 32181928
허찬용 32184939

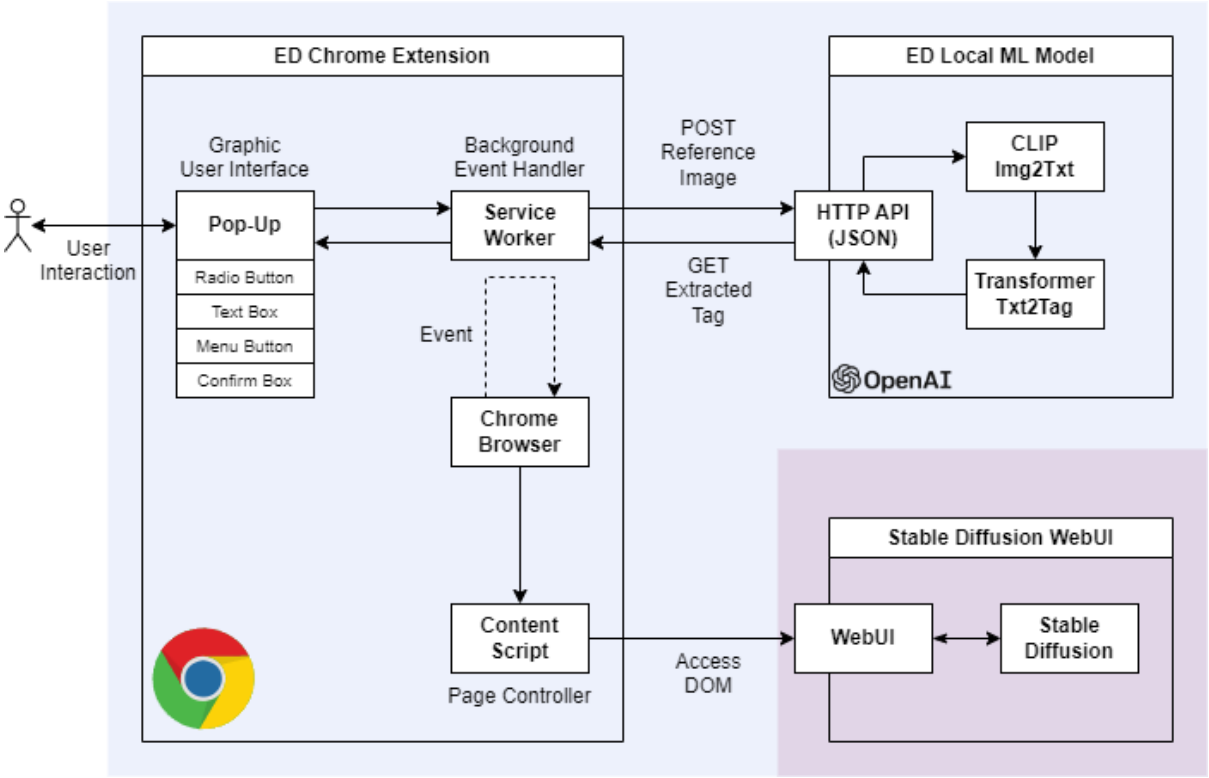
Easy Diffuser

https://github.com/Easy-Diffuser

개요

Easy Diffuser 는 비전문가 사용자가 Stable Diffusion 을 사용하여 이미지를 생성하는 것을 돕는 서비스이다. 본 서비스는 Stable Diffusion 의 생성 조건 탐색을 보조하기 위해 Reference 이미지를 사용한다. 이는 Reference 이미지로부터 생성 조건을 추출하는 기능과 Reference 이미지를 입력하여 img2img 생성을 수행하는 기능으로 구성되어 있다.

전체 구조

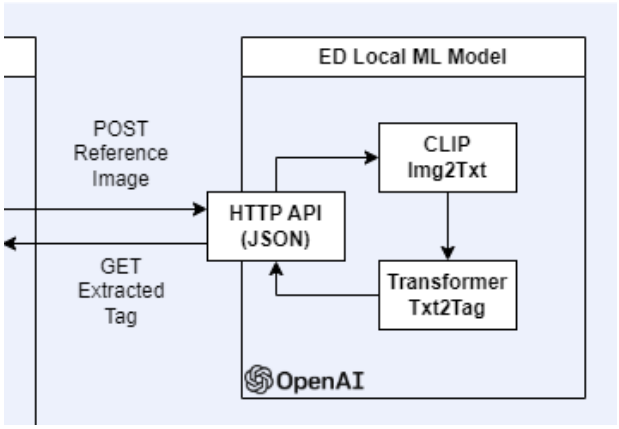


본 서비스는 위와 같은 구조를 통해 Reference 를 통한 이미지 생성을 지원한다. 먼저, 사용자의 편의를 위해 Chrome Extension 을 사용하여 Reference 이미지를 선택하고 불러올 수 있게 한다. 불러온 이미지는 사전학습 된 CLIP 을 통해 이미지를 잘 설명하는 text 로 변환하고, Text 는 다시 Transformer 모델을 거쳐 Tag 로 번역한다. 이러한 결과는 다시 Chrome Extension 을 통해 Stable Diffusion WebUI 로 전송할 수 있다.

세부 구조

Local ML Model

Local ML Model 은 입력된 이미지를 바탕으로 Image-to-text, Text-to-tag 의 두 과정을 거쳐 이미지로부터 Tag 를 추출한다.



Local ML Model 이 외부와 연결되는 API 는 HTTP API 를 사용한다.

1. 외부로부터 이미지를 전달받을 때 http request method는 POST를 사용하고, 아래와 같이 multipart/form-data로 이미지를 전달받는다.

```
<form action="/file/upload" method="post" enctype="multipart/form-data">
    파일명 : <input type="file" name="imagefile">
    <button type="extract">Extract</button>
</form>
```

이때 response 로는 http status code 와 함께 Request body 를 반환한다. 요청이 정상일 경우, 200 을 status code 로 하고 추출 결과를 JSON 으로 반환한다. Exception 이 발생할 경우, 해당하는 http status code 와 함께 Error 시의 message 를 body 로 반환한다. 그 목록은 아래와 같다

Error Message	설명	Status Code
Not an Image File	이미지 파일이 아닌 input 이나 헤더가 전달된 경우	400
Image Invalid	크기가 너무 크거나, 작거나, 이미지의 형식이 지원하지 않는 형식일 경우	415
Unexpected Error	예외처리에 포함되지 않은 에러가 발생한 경우	500

2. 외부로 tag를 반환할 때에는 아래와 같은 JSON을 반환한다.

```
{
    "Tag":[
        "cat",
        "table",
        "realistic"
    ]
}
```

Local ML Model 의 각 component 인 Image-to-text model 과 Transformer 는 pretrained model 로 개별 Package 로 존재한다. 하나의 python 스크립트 내부에서 각 패키지를 불러와 연결하는 방식으로 Local ML Model 을 구현한다.

각 component 의 사용 방식은 아래와 같다

1. Image-to-text

   Image-to-text는 Clip-interrogator 또는 Leeyunjai/img2txt 를 사용할 수 있는데, 둘 모두 별도의 parameter없이 이미지를 load할 수 있다.

   A. Clip-interrogator

      https://huggingface.co/spaces/pharma/CLIP-Interrogator/blob/main/app.py

      ```
      ci = Interrogator(Config(clip_model_name="ViT-L-14/openai"))
      ci.interrogate(image)
      ```

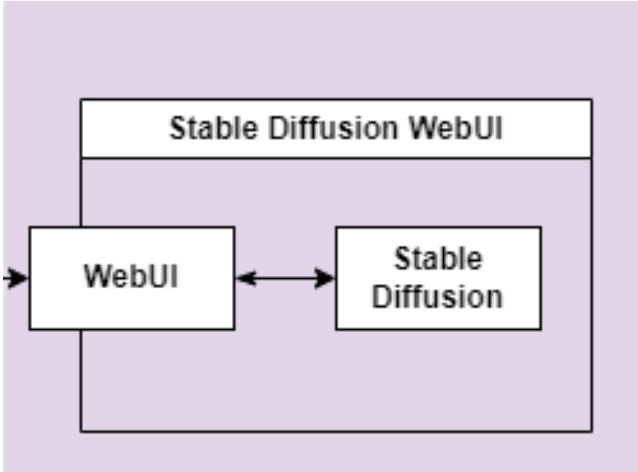
   B. Leeyunjai/img2txt

      ```
      result = model.predict("./image.jpg")
      ```

2. Transformer

   Transformer는 학습에 따라 입력 parameter가 달라질 수 있다.

WebUI



WebUI 는 실제 구현이 필요한 부분은 아니지만, 추출된 조건을 검증하고 구현 결과물을 연결하는 데에 있어 필수적인 요소이다. 특히, 본 서비스의 구현 과정에서 추출 조건의 검증을 위해 이미지 생성을 자동화할 예정인데, 이때 사용되는 방식은 아래와 같다.

먼저 자동화 생성 과정에서는 sdapi/v1/txt2img api 를 사용한다. 이는 Stable Diffusion v1 을 사용하여 text to image 를 수행하는 api 이다.

이를 사용하기 위해서, 먼저 url 변수를 http://127.0.0.1:7860/ 로 고정시켜야 한다. 이는 Local ML model 이 고정된 url 을 통해 WebUI 와 통신하기 위함이다. 특별한 세팅을 하지 않았을 경우 url 은 고정된 상태이다.

Local ML model 에서 생성한 tag 는 JSON 으로 반환되는데, 이때 반환하는 method 를 아래의 호출로 바꾼다.

```
requests.post(url=f'{url}/sdapi/v1/txt2img', json=payload)
```

이를 통해서 payload 라는 dictionary 변수를 txt2img 에 보내고, WebUI 는 payload 를 받아 img 를 생성한다. 생성된 image 는 Return 을 통해서 response 에 담겨 반환된다.

Payload 는 다양한 파라미터를 지원하는데, 자동화 과정에서 주로 변경할 파라미터는

{"prompt": "",
"styles": ["string"],
"sampler_name": "string",
"negative_prompt": "string",
"sampler_index": "Euler",
"script_name": "string"}

이다.

반환은 r=response.json()을 통해 response 를 json 형식으로 받을 수 있다.


```
{'images': ['iVBORw0KGgoAAAANSUhEUgAAAgAAAAIACAIAAAB7GkOtAAAAs3RFWHRwYXJhbWV0ZXJzAGNhdCBkb2cKU3RlcHM6IDUsIFNhbXBsZXI6IEV1bGVyLCBDRkcgc2NhbGU6IDcuMCwgU2VlZDogMjE1NzM0O
gZTA0YjAyMDAxMiwgTW9kZWw6IHJwZ19WNCwgU2VlZCByZXNpemUgZnJvbTogLTF4LTEsIERlbm9pc2luZyBzdHJlbmd0aDogMHhJs+MAAQAASURBVHicbP1rkxxLkiSKqZp5ZBaA7pne3bnclUvKpQj//7/iCne2H6fPA
SgDkgSnI3AiSgMNKNZoRkRiPcTJIZCREyM0AkSUDyYQSMgETQjCTy4yTN6voGACJE0mg0EnKDWV5JXq+FOyGYmRnMQKjeDrjDnQDqOkZzGmFOSmZgfTvMjIQ5mM9CmdHc8qFAkDTPZwAJkjTSDAD7B1ZvzP+wxohEfhEBS
8fPPtq29ffXwd483Gm/ndxm2MGzjMN9qAYOZADgMkCZrngTljnvN4UPM8n4iDOGM+oJOcitMspCkFDYRgqB8zAGYEqXo6A3KU8k35HAaw/0mtv9Ek1Qj88inkgwMGIcdQygGyfC9QvwFEq7dBzJfqH6RQ0wRSETm++VvkQ
0LpzvqJGABLJHo55E13cAhHQ9Xn9TPwiAdekesZD6s8wX+y11G3VXdR+cAmhTDPKUBe2AH7ATNmkTNlwPICAEo0khEIREIaeorpyPp2swFSEaFbXQpZCumc3bW59M3FDol1uV0H+/PjUjn6peCYG4viuxJgI5nhKAkHKaA
i5hNHP2nMmY/RN6AchLpJKWb0o0mSAEXkelCEpJzAmCLzMSXVbQDISciPRoS5xTlrWvNyEIHIv0GInMMa57yl3CAClHc1Q4QijAAYACfGcMg1MIQgYA2ENEDYhiFgwyAZ6A6CMcOdZgZpDJsz3Gp7uNNIQZTMLQd9DMuZo
```


반환된 이미지는 image = Image.open(io.BytesIO(base64.b64decode(i.split(",",1)[0]))) 라는 코드를 통해 이미지 파일로 변환할 수 있다.
io.BytesIO 는 바이트 배열을 이진 배열로 바꾸고, base64.b64decode(i.split(",",1)[0])은 base64 로 인코딩된 이미지 데이터를 디코딩한다. i.split(",",1)[0]는 base64 문자열에서 ',' 이전의 문자열만 추출한다.

이러한 과정을 예시 코드로 나타내면 아래와 같다.

```
import json
import requests
import io
import base64
from PIL import Image, PngImagePlugin

url = "http://127.0.0.1:7860"

payload = {
    "prompt": "puppy dog",
    "steps": 5
}

response = requests.post(url=f'{url}/sdapi/v1/txt2img', json=payload)

r = response.json()

for i in r['images']:
    image = Image.open(io.BytesIO(base64.b64decode(i.split(",",1)[0])))

    png_payload = {
        "image": "" + i
    }
    response2 = requests.post(url=f'{url}/sdapi/v1/png-info', json=png_payload)

    pnginfo = PngImagePlugin.PngInfo()
    pnginfo.add_text("parameters", response2.json().get("info"))
    image.save('output.png', pnginfo=pnginfo)
```

png_payload 는 png-info api 를 위한 json 형식의 변수이고, response2 = requests.post(url=f'{url}/sdapi/v1/png-info', json=png_payload) 를 통해 아래와 같은 response 를 받을 수 있다.

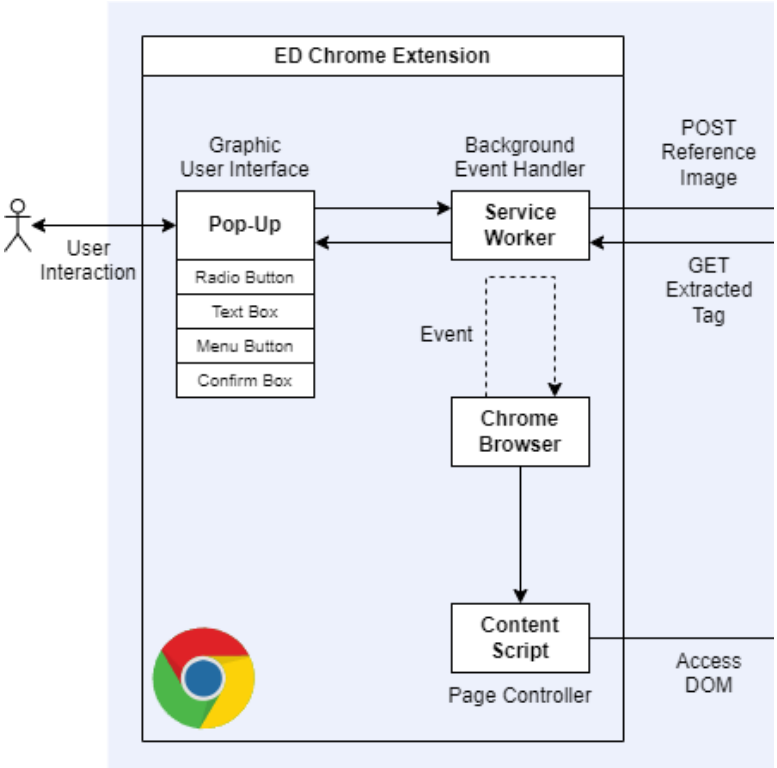
```
{
  "info": "string",
  "items": {}
}
```

아래는 Txt2img api json 의 형식이다.

```
{
  "enable_hr": false,
  "denoising_strength": 0,
  "firstphase_width": 0,
  "firstphase_height": 0,
  "hr_scale": 2,
  "hr_upscaler": "string",
  "hr_second_pass_steps": 0,
  "hr_resize_x": 0,
  "hr_resize_y": 0,
  "prompt": "",
  "styles": [
    "string"
  ],
  "seed": -1,
  "subseed": -1,
  "subseed_strength": 0,
  "seed_resize_from_h": -1,
  "seed_resize_from_w": -1,
  "sampler_name": "string",
  "batch_size": 1,
  "n_iter": 1,
  "steps": 50,
  "cfg_scale": 7,
  "width": 512,
```

```
  "height": 512,
  "restore_faces": false,
  "tiling": false,
  "do_not_save_samples": false,
  "do_not_save_grid": false,
  "negative_prompt": "string",
  "eta": 0,
  "s_churn": 0,
  "s_tmax": 0,
  "s_tmin": 0,
  "s_noise": 1,
  "override_settings": {},
  "override_settings_restore_afterwards": true,
  "script_args": [],
  "sampler_index": "Euler",
  "script_name": "string",
  "send_images": true,
  "save_images": false,
  "alwayson_scripts": {}
}
```

Chrome Extension



Chrome Extension 은 유저와의 상호작용을 처리하며, 동시에 Local ML model 과 WebUI 사이를 연결하는 메인 함수 역할을 수행한다. Chrome Extension 의 component 는 아래와 같다.

1. Pop-up.js 는 라디오 버튼, 텍스트 박스 등의 UI Element를 다룬다. 실제 사용자와의 상호작용이 일어나는 부분이다.

   일반적인 Chrome Extension은 pop-up 시에 popup.html을 호출하여 pop-up 화면을 띄운다. 그러나 Easy Diffuser는 popup.html에서 pop-up을 띄우지 않고, 우측 사이드바를 구성하도록 한다. 그리고 popup.html은 pop-up.js를 통해 chrome api로 사이드바를 연다.

   Pop-up은 아래의 예시 코드와 같이 동적으로 popup을 호출할 수 있게 한다.

```
chrome.storage.local.get('signed_in', (data) => {
    if (data.signed_in) {
        chrome.action.setPopup({popup: 'popup.html'});
    } else {
        chrome.action.setPopup({popup: 'popup_sign_in.html'});
    } });
```

2. Service Worker는 백그라운드에서 이벤트 핸들러로 동작한다. 이벤트 발생 시 크롬 api를 사용하여 브라우저 조작한다. Easy Diffuser에서는 Pop-up.js로부터 user input에 의한 이벤트를 받아 http api 호출 등의 처리를 수행한다. Service Worker는 아래의 코드와 같이 비동기 함수를 chrome의 runtime에 추가하는 방식으로 동작한다.

```
// background.js
chrome.runtime.onInstalled.addListener(async () => {
  for (let [tld, locale] of Object.entries(tldLocales)) {
    chrome.contextMenus.create({
      id: tld,
      title: locale,
      type: 'normal',
      contexts: ['selection'],
    });
  }
});
```

3. Content Script는 페이지 컨트롤러 역할을 수행한다. Content Script는 DOM을 이용하여 페이지를 직접 조작할 수 있다. 따라서 Easy Diffuser에서는 Content Script를 사용하여 WebUI를 직접 조작하고 이를 통해 이미지, 텍스트 등을 추가할 수 있다.