

Easy Diffuser Design

박찬호, 허찬용

목차

1. Architecture
2. Tech Stack
3. Implementation Plan

Architecture

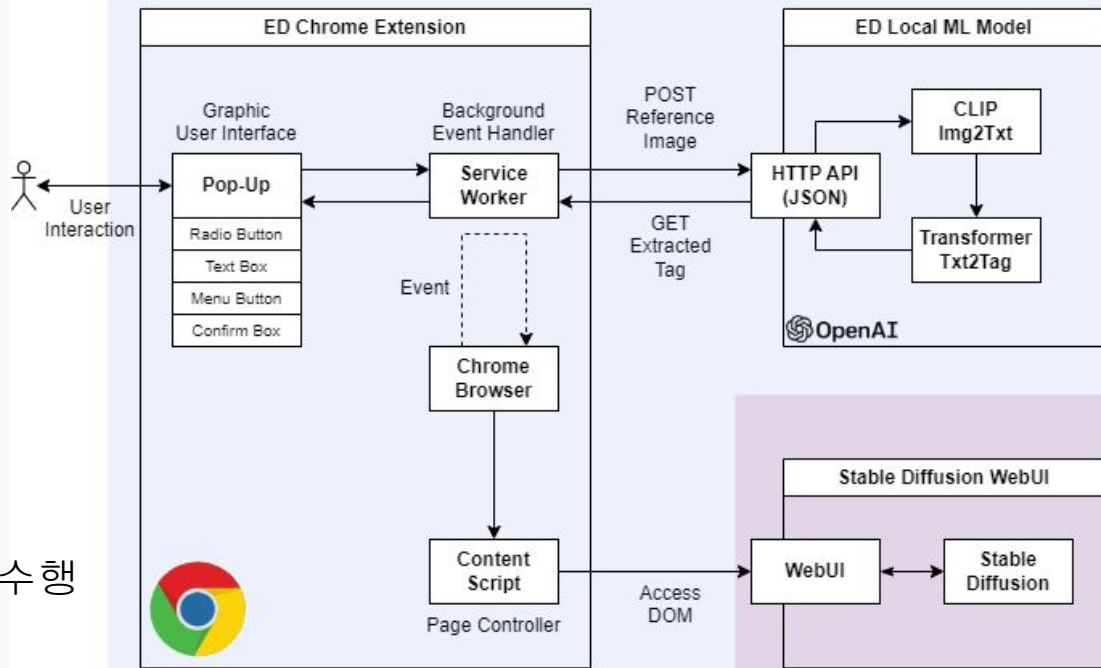
두 부분으로 구성

: Local ML Model

- Img2Txt → Txt2Tag
- 이미지로부터 태그 추출

: Chrome Extension

- 메인 함수 역할
- 유저와의 상호작용을 처리
- UI, Event handling
- 결과물을 WebUI로 보내는 역할도 수행



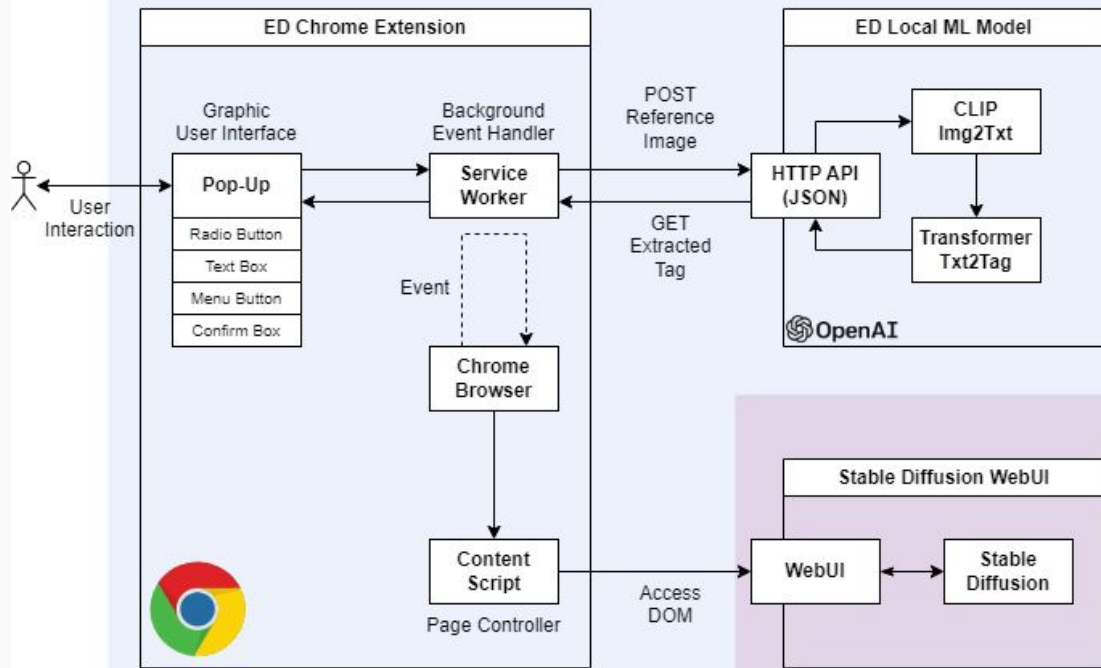
Architecture

기존 문제점

→ 유저가 **WebUI**에서 반복
시행으로 원하는 조건 탐색

Easy Diffuser의 구조

→ Local ML model과 Chrome
Extension이 사이에 끼어들어
시행착오 개선



Tech Stack

1. Chrome Extension
2. CLIP
3. Transformer
4. WebUI

Chrome Extension

- **Pop-up.js : UI Element**
 - 라디오 버튼, 텍스트 박스 등
 - 사용자와의 상호작용
- **Service Worker : 백그라운드 이벤트 핸들러**
 - 이벤트 발생 시 크롬 **api** 사용하여 브라우저 조작
 - Pop-up.js로부터 **user input**에 의한 이벤트를 받아 **http api** 호출 등의 처리 수행
- **Content Script : 페이지 컨트롤러**
 - DOM을 이용하여 페이지 직접 조작 가능
 - WebUI를 직접 조작하여 이미지, 텍스트 등의 추가 기능

CLIP

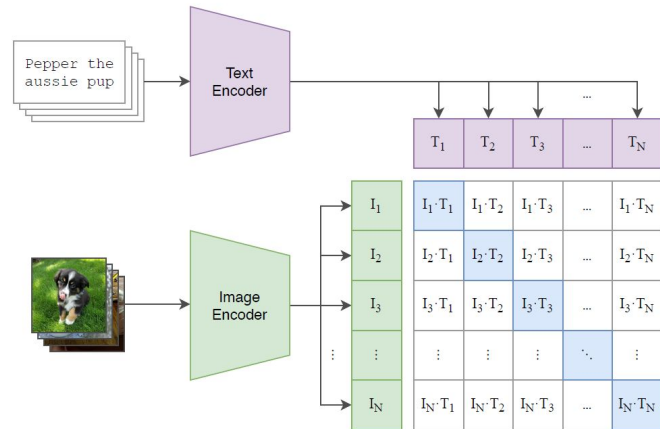
CLIP : OpenAI에서 개발

- 자연어를 supervision으로 하여 Zero-shot Transfer Learning이 가능하게 함

→ Upstream Task로 이미지를 이해 → Downstream Task에 zero shot으로 적용

- Image Encoder 와 Text Encoder를 pair로 하여 joint하게 학습
- Image Encoder – 이미지를 저차원으로 embedding (ResNet-50, ViT)
- Text – Transformer 사용

(1) Contrastive pre-training



CLIP

- Reference를 입력 시 inference 가능
- A photo of a {label}, a type of pet.
- Text captioning에 사용할 수 있다. (실제 학습도 text captioning으로 진행)
- Text Captioning 특화 모델도 있는데 굳이 CLIP 쓰는 이유?
 1. Txt2Tag를 별도 학습하지 않고, CLIP fine tuning으로 해결할 수 있는 가능성이 있어서
 2. 단순 captioning이 아닌 더 다양한 형태의 text output을 만들어볼 수 있어서

→ 이미지를 CLIP에 거쳐 나온 결과물로 Tag 생성

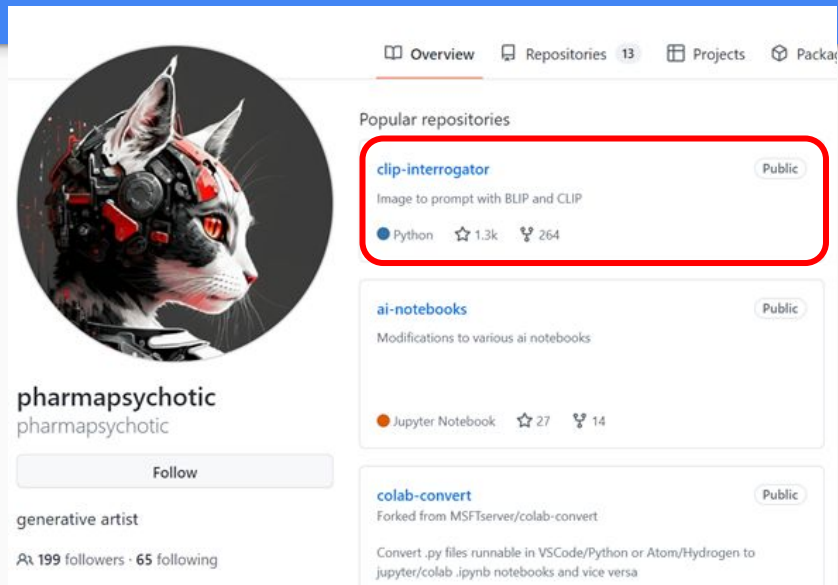
Img2Txt 모델

CLIP(Connecting Text and Images) Model

이미지 삽입 → Script 추출

Model Load : 6초

Model Execution: 1분 10초













Img2Txt 모델




CLIP(Connecting Text and Images) Model

이미지 삽입 → Script 추출

Model Execution: 10초


leeyunjai / **img2txt**   like 2


 Image-to-Text  Transformers  flickr30k  coco2017  English  bert  caption  img2txt

 **Model card**  Files and versions  Community


pretrained model using coco2017 + flickr30k caption dataset

Result

 Image



```
{
  type: "caption",
  result: "ok",
  data: {
    caption: "소파, 의자, 테이블 및 TV가 있는 거실",
    caption_en: "a living room with a couch, chair, table and television"
  }
}
```

 copy to clipboard

Img2Txt 모델 비교

pharmapsychotic/clip-interrogator

결과: Someone feeding a cat with a toothbrush outside on a deck.

Total cost time : 70 Sec

leeyunjai/img2txt 모델

결과: A person feeding a cat with a toothbrush.

Total cost time : 10 Sec



Test 결과

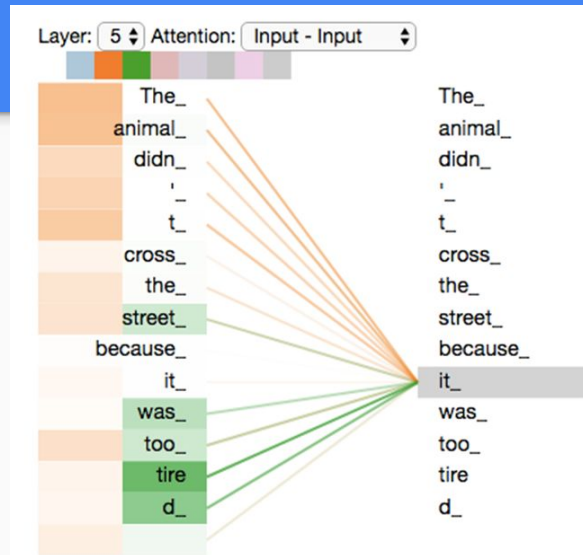
긍정적인 결과

1. Inference 시간이 길지 않음
2. 최대 1분 → 기존의 시행착오 비해 상당한 시간 단축
3. Text Captioning 결과도 준수

남은 과제

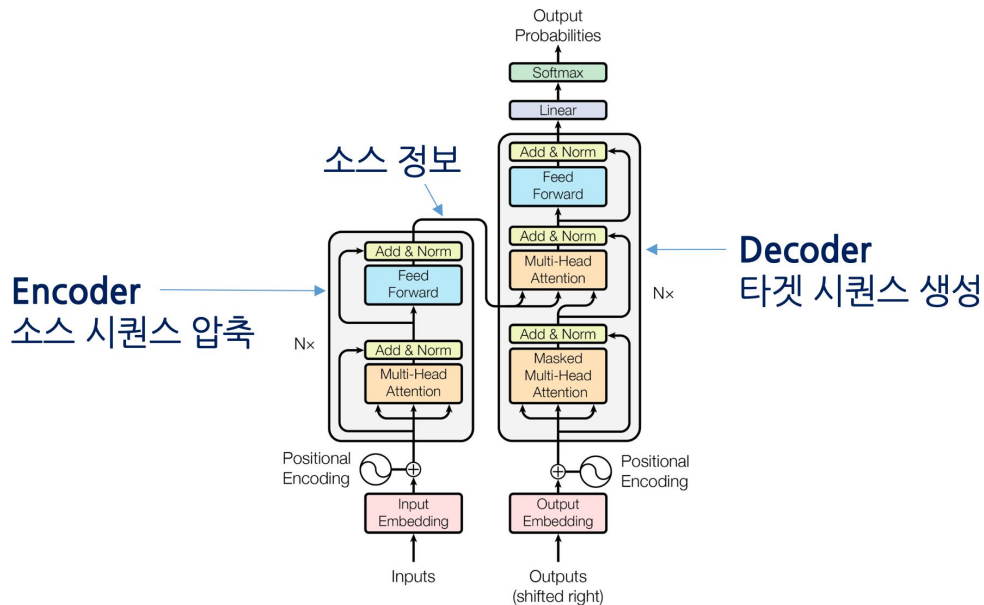
1. Txt2Tag로 Tag를 잘 생성할 수 있을까
2. Captioning은 이미지를 잘 설명하는 Positive Tag와 연관. 실제 Diffusion 생성 시에는 Negative도 중요. Negative Tag를 어떻게 정할 수 있을까

→ Multihead Attention에서 Positive와 정반대되는 값을 출력?



Text2Tag 모델

- Pretrained Transformer 모델 사용
- Script를 넣어 Attention 값을 사용하여 중요도에 대한 가중치를 계산 뒤 임계치를 넘는 단어들을 추출
- Txt2Tag 번역을 수행



WEBUI API

- <http://127.0.0.1:7860/docs> 해당 웹에 있는 api를 통해서 WEBUI를 사용
- JSON 파일을 사용하여 특정 기능 수행
- 생성된 이미지를 로컬 환경에 저장 가능
- Transformer로 얻은 Tag들로 이미지 자동생성 후 성능 확인

POST

/sdapi/v1/img2img Img2Imgapi

Parameters

No parameters

Request body ^{required}

Example Value | Schema

```
{
  "init_images": [
    "string"
  ],
  "resize_mode": 0,
  "denoising_strength": 0.75,
  "image_cfg_scale": 0,
  "mask": "string",
  "mask_blur": 4,
  "inpainting_fill": 0,
  "inpaint_full_res": true,
  "inpaint_full_res_padding": 0,
  "inpainting_mask_invert": 0,
  "initial_noise_multiplier": 0,
  "prompt": "",
  "styles": [
    "string"
  ],
  "seed": -1,
  "subseed": -1,
  "subseed_strength": 0,
  "seed_resize_from_h": -1,
  "seed_resize_from_w": -1,
  "sampler_name": "string",
```

Implementation plan

[illegible]

Implementation plan

- Chrome Extension 개발 → 금일 착수
- Tag 추출 모델 개발 → 개발 양 보다는 학습 시간 많이 소모
 - : Img2Text 모델 개발 → CLIP
 - : Text2Tag 모델 개발 → Transformer
 - 시험 기간 내내 학습 및 추론
- 결과 Tag 출력 기능 개발 → Negative tag 해결 필요

전체 과정 자동화 : 원본 Image → Img2Txt → Txt2Tag → Diffusion 이미지
→ 최종 결과눈으로 비교

감사합니다.