OpenAI 이미지와 음성

DALL·E

DALL·E에 대해

- 2023년 9월, OpenAI는 달리3(DALL·E 3)를 출시
- 이 AI는 기존 인공지능 모델들과 비교할 때 프롬프트를 이해하는 능력이 월등히 뛰어나며, 텍스트를 바탕으로 이미지를 창조하는 기능을 가진 세계적으로 몇 안 되는 모델 중 하나
- 2023년 10월, 달리3는 챗GPT Plus(유료 버전)에 통합되어 챗GPT 대화창을 통해 사용
- Microsoft Copilot 및 Bing Image Creator(무료)에서도 이용 가능
- 이러한 플랫폼들은 사용자가 제공하는 텍스트 프롬프트를 기반으로 상세하고 독창적인 이미지를 생성할 수 있는 능력을 제공

https://labs.openai.com/

DALL·E

DALL·E에 대해

- 달리3가 영어뿐만 아니라 여러 다른 언어를 이해 할 수 있다는 것
- 한국어로 작성된 프롬프트(명령어) 또한 높은 수준으로 잘 이해하며 처리
- 웹사이트에서 이미지를 업로드 하거나, 이미지를 생성하는 기능 구현이 잘 되어 있음
- in-painting, out-painting 도구도 함께 제공

https://labs.openai.com/

DALL·E

이미지 생성

- model
  - 생성: 사용 가능한 모델은 dall-e-3 및 dall-e-2
  - 편집(edit): dall-e-2
  - 기준 이미지에 대한 다양화(variation): dall-e-2
- 이미지 크기
  - dall-e-2
    - 256x256, 512x512, 1024x1024
  - dall-e-3
    - 1024x1024, 1792x1024, 1024x1792

https://platform.openai.com/docs/guides/images/introduction?context=node&api-mode=chat

DALL·E

이미지 생성

- 사진요청 갯수
  - DALL·E 3을 사용하면 한 번에 1개의 이미지를 요청 할 수 있고(병렬 요청을 통해 더 많은 이미지를 요청할 수 있음)
  - DALL·E 2를 사용하면 n 매개변수와 함께 한 번에 최대 10개의 이미지를 요청

https://platform.openai.com/docs/guides/images/introduction?context=node&api-mode=chat

DALL·E

이미지 생성

- Jupyter Notebook 에서 출력하기 위하여 Image 를 사용

```
from IPython.display import Image

# 생성된 이미지를 출력합니다.

Image(url=image_url)
```

- urllib 라이브러리를 사용하여 이미지를 다운로드

```
import urllib

# 생성된 이미지를 URL로부터 다운로드하여 저장합니다.

urllib.request.urlretrieve(image_url, "generated_image.jpg")
```

DALL·E

이미지 생성

- 이미지 생성

```python
import os
from openai import OpenAI
from dotenv import load_dotenv
load_dotenv(override=True)

OPENAI_API_KEY = os.getenv("OPENAI_API_KEY")

# OpenAI 클라이언트 초기화
client = OpenAI()

kwargs = {
    "prompt": "A beautiful landscape."
}

# 이미지 생성
im = client.images.generate(**kwargs)
print(im)
```

DALL·E

이미지 생성

- 다른 크기의 이미지 생성

```
from IPython.display import Image, display

prompt = "A beautiful landscape."
n = 1
size = "256x256"

kwargs = {
    "prompt" : prompt,
    "n" : n,
    "size" : size,
}

im = client.images.generate(**kwargs)

for i in range(n):
    print(im.data[i].url)
    display(Image(url=im.data[i].url))
```

DALL·E

여러 이미지 생성

- 여러 이미지 생성

```
from IPython.display import Image, display

prompt = "A beautiful landscape."
n = 3
size = "512x512"
kwargs = {
    "prompt": prompt,
    "n": n,
    "size" : size,
}

im = client.images.generate(**kwargs)
print(im)

for i in range(n):
    print(im.data[i].url)
    display(Image(url=im.data[i].url))
```

DALL·E

이미지 프롬프트 개선 방법

- 작가 모방하기

```
prompt = "cute smiling dog by Akira Toriyama"
n = 1
size = "512x512"
kwargs = {
    "prompt": prompt,
    "n": n,
    "size": size
}

im = client.images.generate(**kwargs)
print(im)

for i in range(n):
    print(im.data[i].url)
    display(Image(url=im.data[i].url))
```

DALL·E

다양한 버전의 이미지 생성(Image Variation)

- image: 변형의 기준으로 사용할 이미지
- 4MB 미만의 유효한 PNG 파일이어야 하며 정사각형
- model: 이미지 생성에 사용할 모델입니다. 현재 dall-e-2 만 지원
- n: 생성할 이미지의 개수
- size: 256x256, 512x512, or 1024x1024. 기본값은 1024x1024

DALL·E

다양한 버전의 이미지 생성(Image Variation)

- 여러 프롬프트 조합으로 다양한 랜덤 이미지 만들기

```
lists = [colors, resolution, angles, lens, light, filter]

user_prompts = [
    "Happy Darth Vader smiling and waving at tourists in a museum of Star Wars memorabilia.",
    "Darth Vader rapping with 2Pac.",
    "Darth Vader playing the piano.",
    "Darth Vader playing the guitar.",
    "Darth Vader eating sushi.",
    "Darth Vader drinking a glass of milk.",
]

n = 2
```

DALL·E

다양한 버전의 이미지 생성(Image Variation)

- 여러 프롬프트 조합으로 다양한 랜덤 이미지 만들기

```
for user_prompt in user_prompts:
    print(f"\nGenerating images for prompt: {user_prompt}")

    customizations = ""
    for lst in lists:
        if random.choice([True, False]):
            customizations += random.choice(lst) + ", "

    full_prompt = user_prompt + ", " + customizations
    print("Using prompt:", full_prompt)

    im = client.images.generate(
        prompt=full_prompt,
        n=n,
        size=size,
    )

    for idx, img in enumerate(im.data, start=1):
        print(f"Image {idx}")
        display(Image(url=img.url))
        time.sleep(1)

    print("Finished generating images for prompt: " + user_prompt)
```

Whisper API를 사용하여 TTS, STT

Text To Speech(TTS)

- TTS는 컴퓨터 프로그램이나 기기가 텍스트를 인간의 음성처럼 들리는 오디오로 변환하는 과정
- 이 기술은 음성 합성을 통해 텍스트 데이터를 자연스러운 음성으로 바꿈
- 사용 예시: 오디오북, 음성 안내 시스템, 음성 기반 가상 어시스턴트 등

Whisper API를 사용하여 TTS, STT

Text To Speech(TTS)

- model: 사용 가능한 TTS 모델 중 하나를 지정. tts-1 또는 tts-1-hd.
- input: 오디오를 생성할 텍스트입니다. 최대 길이는 4096자
- voice: 오디오를 생성할 때 사용할 음성입니다. 지원되는 음성은 alloy, echo, fable, onyx, nova, and shimmer
- response_format: 오디오를 입력할 형식. 지원되는 형식은 mp3, opus, aac 및 flac
- speed: 생성된 오디오의 속도. 0.25 에서 4.0 사이의 값을 선택.기본값은 1.0

Whisper API를 사용하여 TTS, STT

Text To Speech(TTS)

```
speech_file_path = "tts_audio.mp3"

response = client.audio.speech.create(

    model="tts-1",

    input="아~ 오늘 파이썬 배우기 정말 좋은 날이네~",

    voice="alloy",

    response_format="mp3",

    speed=1.1,

)

response.stream_to_file(speech_file_path)
```

Whisper API를 사용하여 TTS, STT

Text To Speech(TTS)

- 저장한 오디오 파일을 재생

```
from IPython.display import Audio

Audio(speech_file_path)
```

Whisper API를 사용하여 TTS, STT

Speech To Text(STT)

- STT는 사람의 말소리를 텍스트로 변환하는 기술
- 이는 음성 인식을 통해 구어체 언어를 캡처하고 이를 기록 가능한 형태의 텍스트로 변환
- 사용예시: 음성 명령 입력, 자동 회의록 작성, 음성 기반 검색 시스템 등.
- 파일 업로드는 현재 25MB로 제한되어 있으며, 지원되는 입력 파일 형식은 mp3, MP4, MPEG, MPGA, M4A, WAV, WEBM

Whisper API를 사용하여 TTS, STT

Speech To Text(STT)

- file: 변환할 오디오 파일 개체(파일 이름이 아님)로, 다음 형식 중 하나

: FLAC, MP3, MP4, MPEG, MPGA, M4A, OGG, WAV 또는 WEBM.

- model: 현재는 whisper-1 모델만 지정 가능합
- language: 입력 오디오의 언어. 입력 언어를 ISO-639-1 형식으로 제공하면 정확도와 지연 시간이 개선
- prompt: (선택 사항) 모델의 스타일을 안내하거나 이전 오디오 세그먼트를 계속하기 위한 텍스트. 프롬프트는 오디오 언어와 일치
- response_format: 변환된 결과물 출력 형식. 가능한 지정 옵션은 json, text, srt, verbose_json 또는 vtt
- temperature: 0에서 1 사이의 샘플링 temperature

Whisper API를 사용하여 TTS, STT

Speech To Text(STT)

```
audio_file = open("data/채용면접_샘플_01.wav", "rb")

transcript = client.audio.transcriptions.create(
    file=audio_file,
    model="whisper-1",
    language="ko",
    response_format="text",
    temperature=0.0,
)
# 결과물 출력
print(transcript)
```

Whisper API를 사용하여 TTS, STT

더욱 긴 오디오 입력 대한 처리

- 기본적으로 Whisper API는 25MB 미만의 파일 만 지원
- 이보다 긴 오디오 파일이 있는 경우 25MB 이하의 청크로 나누거나 압축된 오디오 형식을 사용
- 최상의 성능을 얻으려면 문장 중간에 오디오를 분할하면 일부 문맥이 손실될 수 있으므로 분할을 피하는 것
- 이를 처리하는 한 가지 방법은 PyDub 오픈 소스 Python 패키지를 사용하여 오디오를 분할 하는 것

Whisper API를 사용하여 TTS, STT

더욱 긴 오디오 입력 대한 처리

```
from pydub import AudioSegment

filename = "data/샘플_02.wav"

myaudio = AudioSegment.from_mp3(filename)

# PyDub 는 밀리초 단위로 시간을 계산합니다.

thirty_seconds = 1 * 30 * 1000  # (1초 * 30) * 1000

total_milliseconds = myaudio.duration_seconds * 1000  # 전체 길이를

밀리초로 변환
```

Whisper API를 사용하여 TTS, STT

더욱 긴 오디오 입력 대한 처리

```
# 전체 길이를 30초로 나누어서 반복할 횟수를 계산합니다.

total_iterations = int(total_milliseconds // thirty_seconds + 1)

total_iteration
```

Whisper API를 사용하여 TTS, STT

더욱 긴 오디오 입력 대한 처리

```
 # 분할된 오디오를 저장합니다.

    part_of_audio.export(output_filename, format="mp3")

    output_filenames.append(output_filename)

# 결과물(파일명) 출력

output_filenames
```

Whisper API를 사용하여 TTS, STT

더욱 긴 오디오 입력 대한 처리

```
 # 생성된 transcript 를 리스트에 추가합니다.

    transcripts.append(transcript)

# 전체 transcript 출력(리스트를 문자열로 변환)

final_output = "---- 분할 ---- ₩n".join(transcripts)

print(final_output)
```

정리

정리

- DALL·E
- Whisper API를 사용하여 TTS, STT