Imaginate API Documents

Imaginate API Documents Table of contents

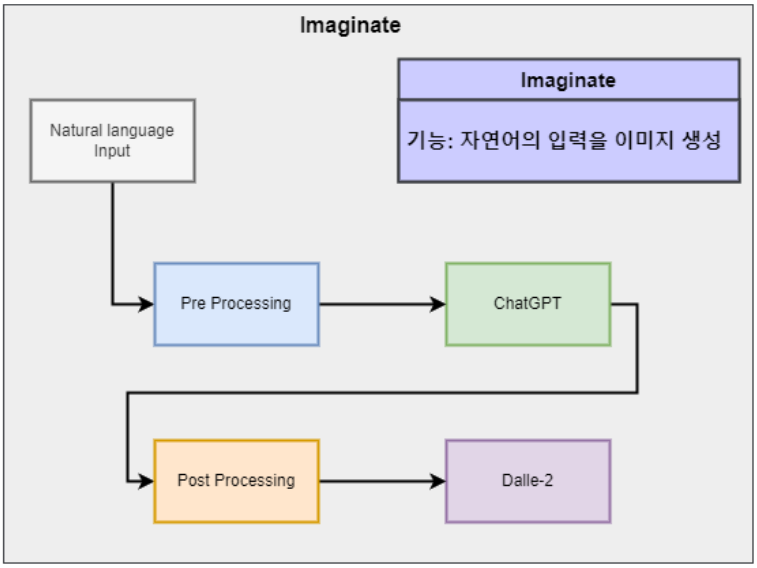
API to create images using dalle2

Dalle 2 Code github

Dalle 2 Test Site

- **API 기능 요약**
- **API 사용 설명서**
- **API 코드 설명**

★API 기능 요약



Imaginate

기능: 자연어의 입력을 이미지 생성

Pre Processing

2개의 경우의 수로 나뉘어서 설정

자연어를 입력한 경우:
간단한 검사 후 리턴

카테고리를 설정해서 입력한 경우:
하나의 문장으로 만들어서 리턴

ChatGPT

영어로 번역 후, 분위기, 주제, 배경, 스타일, 사물 등 카테고리 추출

Post Processing

스타일, 분위기는 정해져 있기에 3개의 방식으로 동의어 처리(이미지에 대한 정보를 저장하기 위해)

1. **datamuse api: datamuse api**를 사용하여 동의어 처리

2. **nltk의 wordnet: nltk의 wordnet** 라이브러리를 사용하여 동의어 처리

3. **fasttext의 cosine similarity** 비교: **cosine similarity**를 비교하여 가장 가까운 동의어로 처리

Dalle-2

Dalle-2 api를 사용하여 이미지 생성

생성하고자 하는 이미지를 카테고리 선택, 자연어 입력을 받아 이미지를 생성하는 API

★API 사용 설명서

2가지의 방법으로 이미지 생성 가능.

카테고리를 선택하여 이미지를 생성하는 경우, 짧은 한 줄 문장으로 변환되어 진행.

3001번 포트를 사용.

	✔openai_query	✔Image_download/<int:number>	✔session_clear
methods	POST	POST	POST

end_point	/openai_query	/Image_download/<int:number>	/session_clear
	POST http://192.168.1.46:3001/openai_query	POST http://192.168.0.45:3001/Image_download/2	
parameters	JSON `{ "moods": "", "theme": "", "background": "", "object": "", "act": "", "styles": "", "number": "4", "size": 512, "search": "따뜻한 바다에서 휴가를 즐기는 여성 사진" }`	URL에 다운로드 받은 number추가	text
Response	`'Completed Dalle Image Download'`	해당하는 number에 대한 이미지	`'Completed Dalle Image Delete'`
기능	카테고리, 자연어 입력을 통한 이미지 생성	생성된 이미지를 다운로드 받기 위함	다운로드된 이미지를 DB에서 삭제하기 위함

★API 코드 설명

- **mood_list, style_list 초기 설정**

해당 리스트는 정해진 값이 존재하기에 이를 벗어난 단어, 동의어들을 처리하기 위한 리스트

```
mood_list = ['exciting', 'joyful', 'sad', 'calm', 'thrilled', 'quiet', 'bright', 'neat', 'modern', 'cute',
             'sophisticated', 'dark', 'vivid', 'light', 'soft', 'calm', 'deep', 'peaceful', 'boring', 'depressed'
             'dynamic', 'frustrating', 'festival', 'romantic', 'passionate', 'noisy', 'chaotic', 'frightening',
             'critical', 'monotonous', 'solitary', 'empty', 'optimistic', 'pessimistic', 'embarrassed', 'warm', '

style_list = ['3D', '2D', 'sketch', 'pastel', 'photography', 'oriental painting', 'western painting', 'neon', 'Go
              'future', 'fantasy', 'close-up', 'aerial shot', 'black and white', 'movie', 'drawing', 'child paint
              'sticker', 'illustration', 'poster', 'character', 'cyberpunk']
```

- **fasttext_model model 설정**

fasttext의 cosine similarity를 계산하기 위한 model

```
fasttext_model = models.fasttext.load_facebook_model(r'D:\fasttext\cc.en.300.bin.gz')
```

- **openai_query_초기 설정**

카테고리, 자연어를 입력했을 때 한 줄 문장으로 수정하기 위한 code

```
# default setting
    number = 4
    size = 256

    # POST type classification
    # paramter를 JSON형식인지 판단 후, 처리
    if flask.request.content_type == 'application/json':
        parameters = dict(flask.request.get_json())
        number = int(parameters['number'])
        size = parameters['size']

```

```
    if not parameters['search']:
        # 사진 -> 고화질 사진으로 번역 처리
        if parameters['styles'] == '사진':
            parameters['styles'] = '고화질 사진'

        # 번역 문장 생성
        if len(parameters['object'].split(',')) > 1:
            object_list = parameters['object'].split(',')
            object_text = '과(와) '.join(object_list)
            text = parameters['moods'] + '분위기의 ' + parameters['theme'] + '주제로 ' + parameters['background']
                  + parameters['act'] + ' ' + object_text + '을(를) ' + parameters['styles'] + '스타일로'
        else:
            text = parameters['moods'] + '분위기의 ' + parameters['theme'] + '주제로 ' + parameters['background']
                  + parameters['act'] + parameters['object'] + '을(를) ' + parameters['styles'] + '스타일로'
    else:
        text = parameters['search']

    text = re.sub(' +', ' ', text)
```

- **openai_query_post processing**

openai api를 호출 후, 해당 결과를 후 처리 하기 위한 code

```
    response_text = response['choices'][0]['text'].lower()
    pattern = r' - '
    response_text = re.sub(pattern, ':', response_text)

    # response에서 mood, theme, background, object, style을 추출하기 위한 부분
    mood_index = response_text.find('mood')
    theme_index = response_text.find('theme')
    background_index = response_text.find('background')
    object_index = response_text.find('object')
    style_index = response_text.find('style')

    # Pattern processing because results are returned in two ways
    pattern = r'mood|theme|background|style|object'

    # 각 mood, theme, background, object, style 패턴을 찾고 해당 값 추출
    mood_val = re.sub(pattern, '', response_text[mood_index:].split(':')[1]).strip()
    theme_val = re.sub(pattern, '', response_text[theme_index:].split(':')[1]).strip()
    background_val = re.sub(pattern, '', response_text[background_index:].split(':')[1]).strip()
    object_val = re.sub(pattern, '', response_text[object_index:].split(':')[1]).strip()
    style_val = re.sub(pattern, '', response_text[style_index:].split(':')[1]).strip()

    if response_text[mood_index:mood_index+4] == 'mood':
        # Create a list of synonyms to use for WordNet
        synonyms = []  # 동의어를 저장할 리스트

        for syn in wordnet.synsets(mood_val):
            for lemma in syn.lemmas():
                synonyms.append(lemma.name())  # lemma.name()은 동의어를 문자열로 반환함

        synonyms = list(set(synonyms))  # set()을 사용하여 중복된 동의어를 제거함

        # Create a list of synonyms to use for datamuse
        # datamuse를 사용해서 동의어 search
```

```
        datamuse_url = f"https://api.datamuse.com/words?rel_syn={mood_val}"
        datamuse_response = requests.get(datamuse_url)
        datamuse_data = datamuse_response.json()

        if datamuse_data:
            datamuse_synonyms = [d["word"] for d in datamuse_data]
        else:
            datamuse_synonyms = []

        # Examine synonyms
        if mood_val not in mood_list:
            # datamuse api synonyms search, step 1
            for i in datamuse_synonyms:
                if i in mood_list:
                    result['moods'] = i
            else:
                # nltk.wordnet synonyms search, step 2
                for i in synonyms:
                    if i in mood_list:
                        result['moods'] = i
                else:
                    # fasttext cosine similarity synonyms search, step 3
                    most_similarity = -1
                    for i in mood_list:
                        if most_similarity < fasttext_model.wv.similarity(mood_val, i):
                            most_similarity = fasttext_model.wv.similarity(mood_val, i)
                            result['moods'] = i
        else:
            result['moods'] = mood_val
```

- **openai_query_dalle 2**

dalle2를 호출하여 이미지를 생성 후, 저장하는 code

```
    # dalle2에 이미지 생성
    response = openai.Image.create(
        prompt= response['choices'][0]['text'], #text[0],
        n=number,
        size=str(size) + 'x' + str(size)
    )

    result['url'] = []

    # 저장할 디렉토리 확인
    if not os.path.exists(dir_path):
        os.makedirs(dir_path, exist_ok=True)

    # 생성된 dalle2링크를 저장
    for idx, res in enumerate(response['data']):
        result['url'].append(res['url'])

        savename = "generated_" + str(idx) + ".png"
        url = result['url'][idx]

        mem = request.urlopen(url).read()
        with open(os.path.join(dir_path, savename), mode="wb") as f:
```

```
            f.write(mem)

    # 생성된 이미지 리스트 생성
    file_list = []
    for i in os.listdir(dir_path):
        file_list.append(os.path.join(dir_path, i))
```

- **Image_download**

Server측에 저장된 이미지를 다운받기 위한 code

```
# 반드시 주소에 다운받을 이미지의 number를 넣어줘야함 (ex. Image_download/0)
@openAI.route('/Image_download/<int:number>', methods=['POST'])
def Image_download(number):
    Image_path = dir_path

    # Server측에 저장되어 있는 이미지를 찾고 해당하는 number의 이미지 반환
    with os.scandir(Image_path) as entries:
        Image_list = [entry.name for entry in entries if entry.is_file() and entry.name.endswith('.png')]
        return send_file(os.path.join(dir_path, Image_list[number]), as_attachment=True)
```

- **session_clear**

Server측에 저장된 이미지를 삭제하기 위한 code

```
def session_clear():
    # parameter를 clear: True/False 형식으로 받아야함
    clear = flask.request.args.get('clear', False)

    if clear:
        session.clear()
        session.modified = True

        file_list = []
        for i in os.listdir(dir_path):
            file_list.append(os.path.join(dir_path, i))
        # 파일 삭제
        for file_name in file_list:
            try:
                # 파일을 닫음
                with open(file_name, 'r') as f:
                    f.close()
                # 파일을 삭제함
                os.remove(file_name)
            except Exception as e:
                print(f"Error while deleting file {file_name}: {e}")
    return 'Completed Dalle Image Delete'
```

- **Run Flask API**

Flask API를 실행하기 위한 code

```python
# ./app.py

CORS(app)
#CORS 서버는 Access-Control-Allow-Origin: *  으로 응답 모든 도메인에서 접근할 수 있음을 의미
cors = CORS(app, resources={r"/api/*": {"origins": "*"}})
if __name__ == "__main__":
    app.run(host='0.0.0.0', port=3001, threaded=True, debug=True, use_reloader=False)
```