

**Московский государственный технический
университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по домашнему заданию
«Image-to-image search с бот интерфейсом»

Выполнил:
студент группы ИУ5-35Б
Никулин И.М.

Подпись и дата:

Проверил:
преподаватель
Гапанюк Ю. В.
Подпись и дата:

Москва, 2024 г

Постановка задачи

Необходимо разработать бота для социальной сети ВКонтакте, который сможет находить похожие изображения на языке python. Пользователь отправляет боту ссылку на изображение или прикрепляет фотографию, а бот возвращает ссылки на похожие изображения из своей базы данных.

Функциональные требования:

Обработка сообщений: Бот должен обрабатывать текстовые сообщения, содержащие URL изображений, а также вложения с фотографиями.

Извлечение URL изображений: Бот должен корректно извлекать URL изображений из текста сообщений и из прикрепленных фотографий.

Поиск похожих изображений: Бот должен использовать методы компьютерного зрения (например, CLIP) и эффективные алгоритмы поиска (например, FAISS) для поиска похожих изображений в базе данных.

Формирование ответа: Бот должен формировать ответ, содержащий ссылки на найденные похожие изображения.

Обработка ошибок: Бот должен корректно обрабатывать ошибки, возникающие при скачивании изображений, генерации эмбеддингов или выполнении поиска, и возвращать информативные сообщения об ошибках.

Хранение данных: Бот должен хранить базу данных изображений (URL и эмбеддинги) и индекс для быстрого поиска.

Использованы библиотека vk_api для взаимодействия с API ВКонтакте, CLIP для генерации эмбеддингов изображений и FAISS для быстрого поиска похожих изображений.

Текст программы

```
import vk_api
from vk_api.longpoll import VkLongPoll, VkEventType
from Bot import VkBot
import random
from config import TOKEN
```

```
# 'https://sun9-43.userapi.com/impf/8Vg_obE4Rjkafg8xWTbhvmkFqn1ZiYxpdCqzA/E0IfNHxcKPU.jpg?size=604x604&quality=96&sign=38e9a2018d1fdb4beedd84755a3fc241&type=album'
```

```
def write_msg(user_id, message):  
    vk.method('messages.send', {  
        'user_id': user_id,  
        'message': message,  
        'random_id': random.randint(1, 2**31)  
    })
```

```
vk = vk_api.VkApi(token=TOKEN)  
longpoll = VkLongPoll(vk)  
bots = {}
```

```
print("Server started")  
for event in longpoll.listen():  
    if event.type == VkEventType.MESSAGE_NEW:  
        if event.to_me:  
            print('Сообщение:')  
            print(f' ID: {event.user_id}', end='')  
            print(' Содержание: ', event.text)  
  
            user_id = event.user_id  
            if user_id not in bots:  
                bots[user_id] = VkBot(user_id)  
  
            bot = bots[user_id]  
            response_message = bot.new_message(event.text,  
event.attachments)  
            write_msg(user_id, response_message)
```

kitties.py

```
import torch
import clip
from PIL import Image
import json
import os
import faiss
import numpy as np
import requests
from io import BytesIO
from config import DB_FILE, FAISS_INDEX_FILE

class ImageSearch:
    def __init__(self, db_file=DB_FILE,
faiss_index_file=FAISS_INDEX_FILE):
        self.db_file = db_file
        self.faiss_index_file = faiss_index_file
        self.device = "cuda" if torch.cuda.is_available() else "cpu"
        self.model, self.preprocess = clip.load("ViT-B/32",
device=self.device)
        self.db = self.load_db()

        if os.path.exists(self.faiss_index_file):
            self.faiss_index = faiss.read_index(self.faiss_index_file)
            self.image_urls = list(self.db.keys())

        else:
            self.faiss_index, self.image_urls =
self.build_faiss_index()

    def load_db(self):
        if os.path.exists(self.db_file):
```

```

        with open(self.db_file, "r") as f:
            return json.load(f)
    return {}

def save_db(self):
    with open(self.db_file, "w") as f:
        json.dump(self.db, f)

def download_image(self, image_url):
    try:
        response = requests.get(image_url, stream=True)
        response.raise_for_status()
        image = Image.open(BytesIO(response.content))
        return image
    except Exception as e:
        print(e)
        return None

def get_image(self, image_url):
    try:
        response = requests.get(image_url, stream=True)
        response.raise_for_status()
        return BytesIO(response.content)
    except requests.exceptions.RequestException as e:
        print(e)
        return None

def get_image_embedding(self, image_url):
    image = self.download_image(image_url)
    if image:
        try:

```

```

        image =
self.preprocess(image).unsqueeze(0).to(self.device)
        with torch.no_grad():
            return
self.model.encode_image(image).cpu().numpy().astype('float32')
        except Exception as e:
            print(f"Error processing image: {e}")
return None

```

```

def add_to_db(self, image_url):
    if image_url in self.db:
        print(f"{image_url} уже есть.")
        return False
    embedding = self.get_image_embedding(image_url)
    if embedding is not None:
        self.db[image_url] = embedding.tolist()
        self.save_db()
        self.faiss_index, self.image_urls =
self.build_faiss_index() # Rebuild index
        print(f"Added {image_url}")
        return True
    else:
        print(f"Ошибка добавления")
        return False

```

```

def build_faiss_index(self):
    embeddings = []
    image_urls = []
    for url, embedding_list in self.db.items():
        embeddings.append(np.array(embedding_list))
        image_urls.append(url)

```

```

        if not embeddings:
            print("KYS")
            return None, []

        embeddings = np.array(embeddings).astype('float32')
        embeddings = embeddings.reshape(len(embeddings), -1)
        index = faiss.IndexFlatL2(embeddings.shape[1])
        index.add(embeddings)
        faiss.write_index(index, self.faiss_index_file)
        return index, image_urls

    def search_similar_images(self, image_url, n=5,
                             similarity_threshold=-10e10):
        query_embedding = self.get_image_embedding(image_url)
        if query_embedding is None:
            return []

        D, I = self.faiss_index.search(query_embedding, n)

        similar_image_urls = []
        for i, distance in zip(I[0], D[0]):
            similarity = 1 - distance / np.sqrt(2)
            if similarity >= similarity_threshold:
                similar_image_urls.append(self.image_urls[i])
        return similar_image_urls

    def search_by_text(self, text, n=5, similarity_threshold=-10e10):
        # Проверяет на наличие текста в итоге
        with torch.no_grad():
            text_input = clip.tokenize([text]).to(self.device)
            query_embedding =
self.model.encode_text(text_input).cpu().numpy().astype('float32')

```

```

if query_embedding is None:
    return []

D, I = self.faiss_index.search(query_embedding, n)

similar_image_urls = []
for i, distance in zip(I[0], D[0]):
    similarity = 1 - distance / np.sqrt(2)
    if similarity >= similarity_threshold:
        similar_image_urls.append(self.image_urls[i])
return similar_image_urls

```

Bot.py

```

from kitties import ImageSearch
from config import DB_FILE, FAISS_INDEX_FILE

class VkBot:
    def __init__(self, user_id):
        self._USER_ID = user_id
        self.image_search = ImageSearch(DB_FILE, FAISS_INDEX_FILE)

    def new_message(self, text, attachments=None):
        image_url = None

        if "http" in text:
            image_url = text.strip()
            image_url = ''.join(image_url.split('&'))

        elif attachments:
            photo_sizes = attachments['photo']['sizes']
            max_res_photo = max(photo_sizes, key=lambda size:
size['width'] * size['height'])
            image_url = max_res_photo['url']

```



```

        # не робит(

        if image_url:
            similar_images =
self.image_search.search_similar_images(image_url=image_url)
            if similar_images:
                response = "Похожие картинки:\n" +
"\n".join(similar_images)
                return response
            else:
                return "Ошиька(."

        else:
            return "Отправьте ссылку на картинку или фото для поиска
похожих изображений."

```

scrape.py

```

from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
from time import sleep
from kitties import ImageSearch

def driver_prep():
    driver = webdriver.Chrome()
    driver.get("https://vk.com/im?sel=c37")

    sleep(15)
    driver.get("https://vk.com/im?sel=c37")
    sleep(30)

    html = driver.find_element(By.TAG_NAME, 'html')

```

```
SCROLL_PAUSE_TIME = 2
```

```
last_height = driver.execute_script("return  
document.body.scrollHeight")
```

```
while True:
```

```
    html.send_keys(Keys.HOME)
```

```
    sleep(SCROLL_PAUSE_TIME)
```

```
    new_height = driver.execute_script("return  
document.body.scrollHeight")
```

```
    if new_height == last_height:
```

```
        break
```

```
    last_height = new_height
```

```
return driver
```

```
def extract_image_urls(driver):
```

```
    image_urls = []
```

```
    pics = driver.find_elements(By.CSS_SELECTOR, '[aria-  
label="фотография"]')
```

```
    for pic in pics:
```

```
        junk = pic.get_attribute('onclick')
```

```
        if junk:
```

```
            junk = junk[junk.find('http'):]
```

```
            junk = junk[:junk.find('")]']
```

```
            junk = junk.replace('\\\\', '/')
```

```
            image_urls.append(junk)
```

```
    return image_urls
```

```
driver = driver_prep()
```

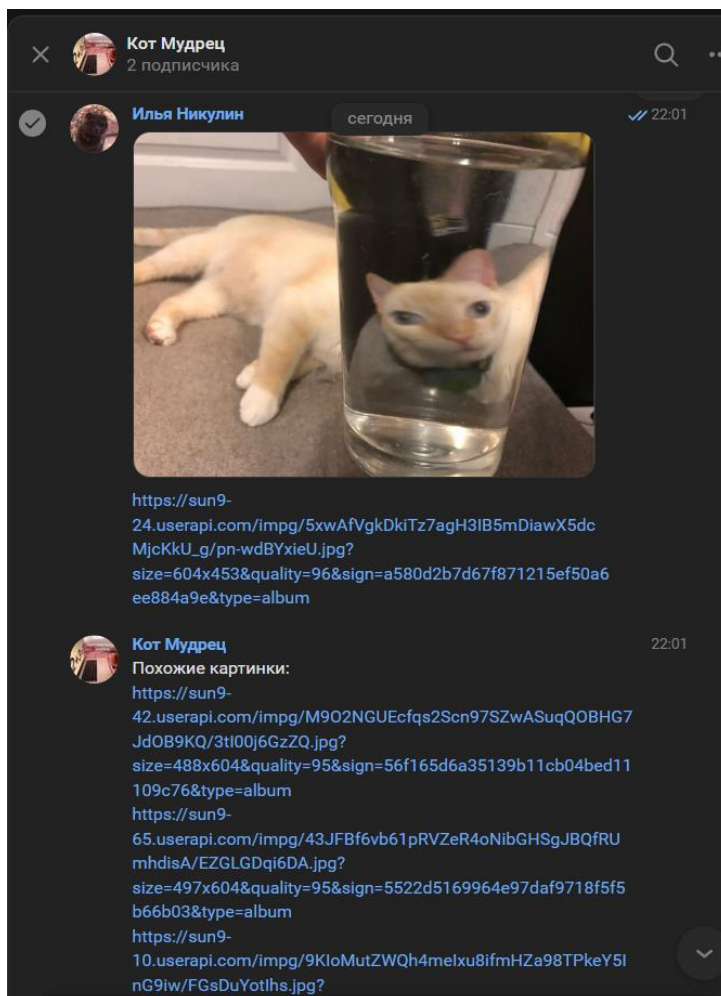
```
image_urls = extract_image_urls(driver)
```

```
print(image_urls)
```

```
for e in image_urls:  
    I = ImageSearch()  
    I.add_to_db(e)
```

Анализ результатов


```
C:\Users\User\PycharmProjects\pop\Kitties\Scripts\python.exe C:\Users\User\PycharmProjects\Kitties\main.py  
Server started  
Сообщение:  
ID: 526980461 Содержание: https://sun9-24.userapi.com/impG/5xwAfVgkDkiTz7agH3IB5mDiawX5dcMjcKkU\_g/pn-wdBYxieU.jpg?size=604x453&quality=96&sign=a580d2b7d67f871215ef50a6ee884a9e&type=album
```





Кот Мудрец
2 подписчика

сегодня



<https://sun9-23.userapi.com/impf/2U5QvUqOiDYJI4UNGoU5t30SMdiRTMKgZFSeWQ/ZSDpFjBlvkQ.jpg?size=604x483&quality=96&sign=501b6972bb4fbd03ae32837513d5e065&type=album>

Кот Мудрец
Похожие картинки:
https://sun9-58.userapi.com/impf/kJLzKMrDC5BohIU0dMB9fJzuN3PVVEP1RaLCAA/OQI_leSiLaE.jpg?size=485x604&quality=96&sign=3c37d7e1ca5fba8c901081a70db1e3fe&type=album
https://sun9-28.userapi.com/impf/txkRGxqC6FdAlBi-uzo-_ZkWG2Z75agObs15w/hCNmhtiX59Y.jpg?size=603x604&quality=95&sign=e0ca321f40dfbd4cd4a7dcc80258748481&type=album

22:13

