

# AI VIET NAM - Tutorial

## Truy vấn ảnh dùng các toán cơ bản

Author: N. D. Tiem, N. V. Quang  
Advisors: N. T. A. Khoa, D. Q. Vinh

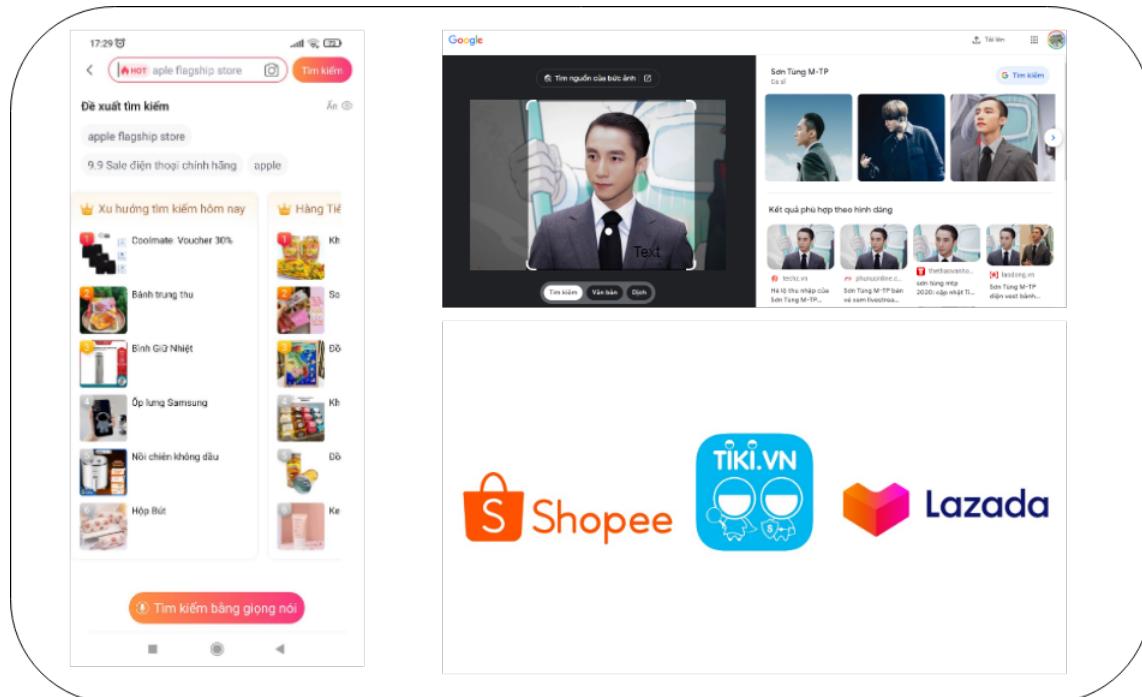
Ngày 8 tháng 8 năm 2023

### Mục lục

<b>1</b>	<b>Giới thiệu</b>	<b>2</b>
1.1	Yêu cầu của bài toán . . . . .	2
1.2	Phân tích và thiết kế hệ thống . . . . .	3
<b>2</b>	<b>Xây dựng chương trình</b>	<b>5</b>
2.1	Thu thập và xử lý dữ liệu . . . . .	5
2.1.1	Thu thập dữ liệu - Crawl data . . . . .	5
2.1.2	Xử lí dữ liệu-Data preprocessing . . . . .	13
2.2	Xây dựng chương trình truy vấn ảnh . . . . .	15
2.2.1	Tính độ tương đồng với độ đo L1 . . . . .	15
2.2.2	Truy vấn hình ảnh với độ đo L2 . . . . .	17
2.2.3	Truy vấn hình ảnh với độ đo Cosine Similarity . . . . .	18
2.2.4	Truy vấn hình ảnh với độ đo Correlation Coefficient . . . . .	18
<b>3</b>	<b>Thử nghiệm và đánh giá kết quả</b>	<b>19</b>
<b>4</b>	<b>Kết Luận</b>	<b>21</b>
<b>5</b>	<b>Phản hồi từ bạn đọc</b>	<b>21</b>

## 1 Giới thiệu

Truy vấn hình ảnh (Images Retrieval) là một bài toán thuộc lĩnh vực Truy vấn thông tin (Information Retrieval). Trong đó, nhiệm vụ của ta là xây dựng một chương trình trả về các hình ảnh (Images) có liên quan đến hình ảnh truy vấn đầu vào (Query) và các hình ảnh được lấy từ một bộ dữ liệu hình ảnh cho trước, hiện nay có một số ứng dụng truy vấn ảnh như: Google Search Image, chức năng tìm kiếm sản phẩm bằng hình ảnh trên Shopee, Lazada, Tiki, ... Trong hướng dẫn này, chúng ta cùng nhau xây dựng một hệ thống truy vấn ảnh cực thú vị nhé.



Hình 1: Một số ứng dụng của Image Retrieval.

Bạn đọc chú ý, điều kiện tiên quyết để đọc bài viết hiệu quả:

- Có kiến thức cơ bản về các độ đo L1, L2, Cosine Similarity, Correlation Coefficient....
- Có kiến thức cơ bản về webcrawl
- Có kiến thức python cơ bản

### 1.1 Yêu cầu của bài toán

Xây dựng một chương trình mà khi đưa vào chương trình đó một hình ảnh, nó sẽ hiển thị nhiều hình ảnh tương tự. Danh sách các loại sự vật mà chương trình có như sau:

```
animal = ["Monkey", "Elephant", "cows",
"Cat", "Dog", "bear", "fox", "Civet",
"Pangolins", "Rabbit", "Bats", "Whale",
"Cock", "Owl", "flamingo", "Lizard", "Turtle",
"Snake", "Frog", "Fish", "shrimp", "Crab", "Snail",
"Coral", "Jellyfish", "Butterfly", "Flies", "Mosquito",
"Ants", "Cockroaches", "Spider", "scorpion", "tiger",
"bird", "horse", "pig", "Alligator", "Alpaca",
```

```

"Anteater", "donkey", "Bee", "Buffalo", "Camel",
"Caterpillar", "Cheetah", "Chicken", "Dragonfly",
"Duck", "panda", "Giraffe"]

plant = ["Bamboo", "Apple", "Apricot", "Banana", "Bean",
"Wildflower", "Flower", "Mushroom", "Weed", "Fern", "Reed",
"Shrub", "Moss", "Grass", "Palmtree", "Corn", "Tulip", "Rose",
"Clove", "Dogwood", "Durian", "Ferns", "Fig", "Flax", "Frangipani",
"Lantana", "Hibiscus", "Bougainvillea", "Pea", "OrchidTree", "RangoonCreeper",
"Jackfruit", "Cottonplant", "Corneliantree", "Coffeeplant", "Coconut"
, "wheat", "watermelon", "radish", "carrot"]

furniture = ["bed", "cabinet", "chair", "chests", "clock",
"desks", "table", "Piano", "Bookcase", "Umbrella", "Clothes",
"cart", "sofa", "ball", "spoon", "Bowl", "fridge", "pan", "book"]

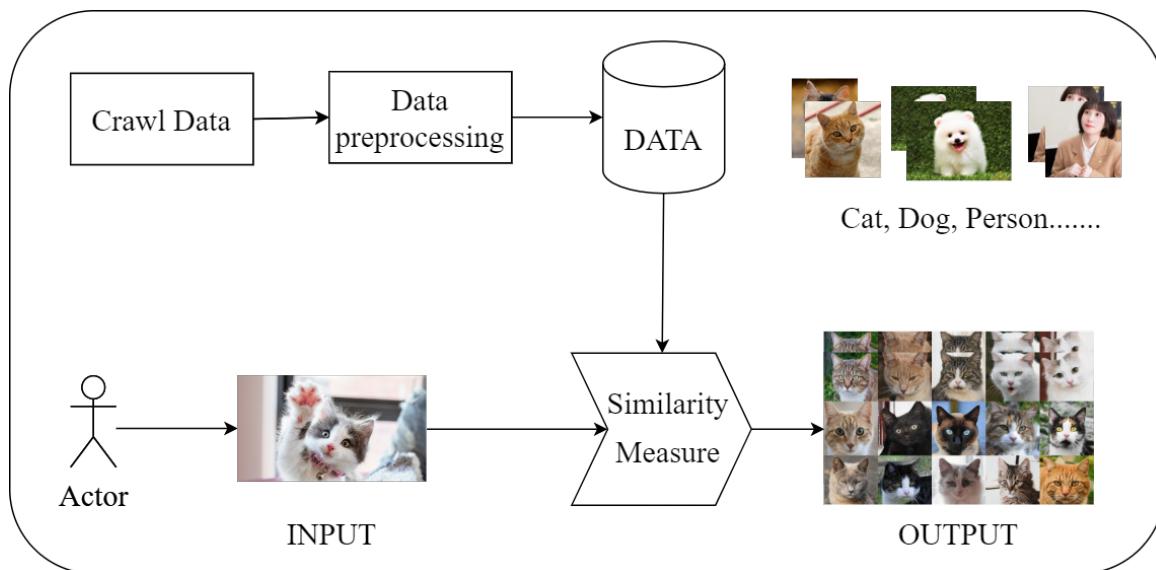
scenery = ["Cliff", "Bay", "Coast", "Mountains", "Forests",
"Waterbodies", "Lake", "desert", "farmland", "river", "hedges",
"plain", "sky", "cave", "cloud", "flowergarden", "glacier",
"grassland", "horizon", "lighthouse", "plateau", "savannah", "valley", "volcano",
"waterfall"]

```

Dựa vào thông tin khảo sát được từ ông hiệu trưởng, chúng ta phân tích và lựa chọn giải pháp truy xuất hình ảnh dựa trên nội dung và tiến hành triển khai dự án.

## 1.2 Phân tích và thiết kế hệ thống

Có rất nhiều cách thiết kế hệ thống truy vấn hình ảnh khác nhau, tuy nhiên về mặt tổng quát sẽ có một pipeline chung sau đây:

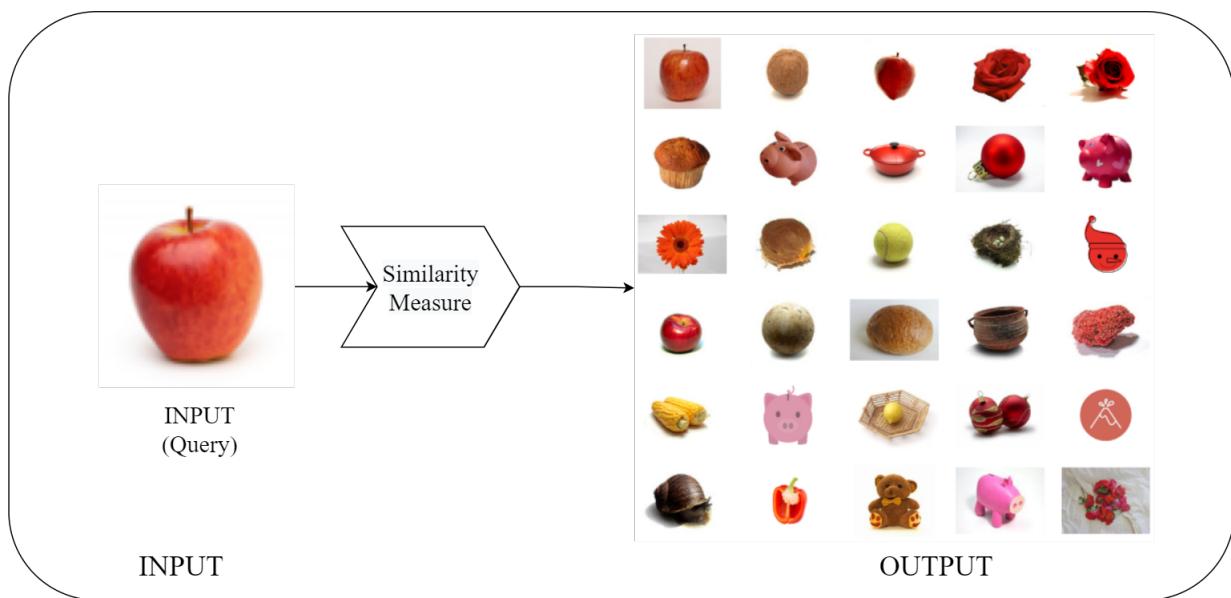


Hình 2: Pipeline tổng quan của một hệ thống Images Retrieval.

Dựa vào hình trên, có thể phát biểu Input/Output của một hệ thống truy vấn văn bản bao gồm:

- **Input:** Hình ảnh truy vấn  $q$  và bộ dữ liệu  $C$ .

- **Output:** Danh sách các hình ảnh  $c$  ( $c \in C$ ) có sự tương quan đến hình ảnh truy vấn.



Hình 3: Demo về hệ thống truy vấn hình ảnh

## 2 Xây dựng chương trình

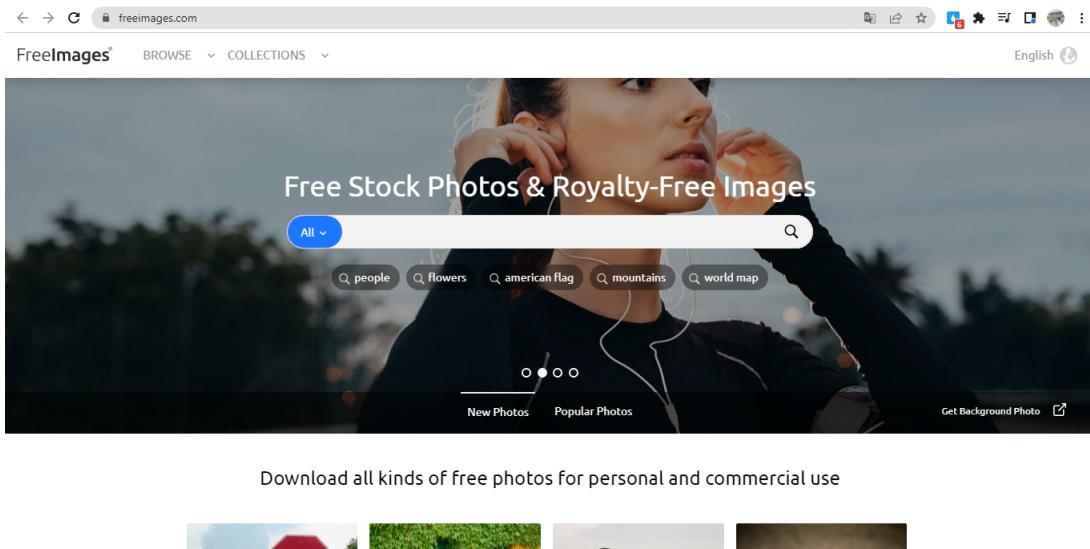
Trong dự án này, chúng ta sẽ xây dựng một chương trình cho phép truy vấn hình ảnh sử dụng các phép đo độ tương đồng giữa các hình ảnh (Similarity Measure). Chúng ta sẽ xây dựng chương trình qua 2 giai đoạn sau:

- Giai đoạn 1: Thu thập (Crawl Data) và xử lý dữ liệu (Data Preprocessing).
- Giai đoạn 2: Xây dựng chương trình truy vấn ảnh.

### 2.1 Thu thập và xử lý dữ liệu

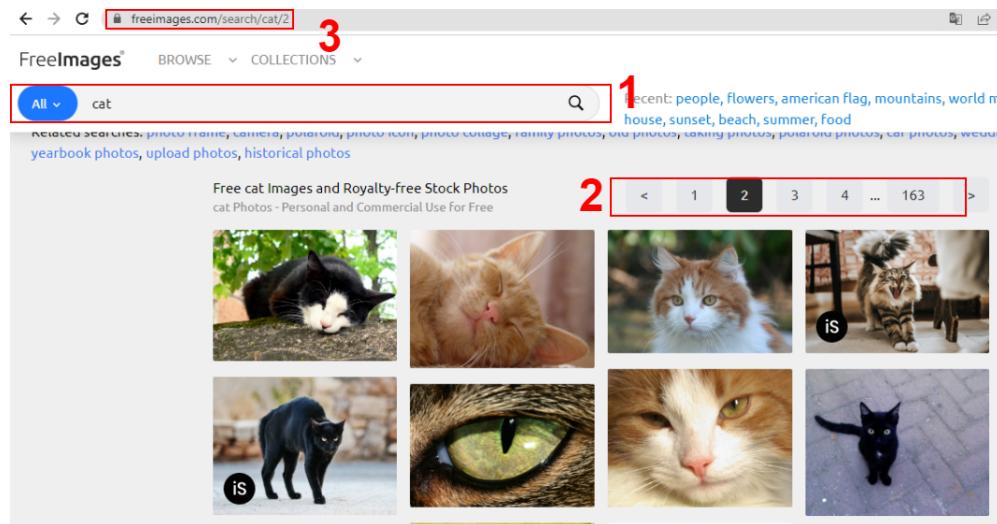
#### 2.1.1 Thu thập dữ liệu - Crawl data

Để xây dựng bộ dữ liệu cho bài toán Images Retrieval việc đầu tiên chúng ta làm là thu thập dữ liệu. Có rất nhiều cách thu thập dữ liệu: sử dụng google tìm kiếm, các thiết bị ghi hình, Image Downloader,... Trong dự án này, chúng ta sẽ thu thập dữ liệu hình ảnh từ trang freeimages.com-một trang chia sẻ các hình ảnh miễn phí với chất lượng cao.



Hình 4: Trang chủ freeimages.com

Chúng ta đã có một vài phương án đề xuất để crawl dữ liệu từ freeimages.com, bây giờ chúng ta sẽ truy cập trang web này và khảo sát cấu trúc cơ bản của trang web này nhé. Chúng ta sẽ sử dụng chức năng tìm kiếm, mình sẽ tìm kiếm con mèo Cat.



Hình 5: Sử dụng chức năng tìm kiếm trên freeimages.com

Ở đây có 3 điểm cần chú ý, (1): Nơi nhập từ khóa tìm kiếm, (2): Số lượng các trang hiển thị kết quả, kết quả sẽ hiển thị trong nhiều trang khác nhau, 1,2,3,4,... (3): Đường dẫn kết quả tìm kiếm. Tương tự bạn tìm con chó, con cừu.....và nhận thấy các đường dẫn có điểm chung là đều bắt đầu với "freeimages.com/search/" chỉ khác nhau bởi từ khóa tìm kiếm và số của trang hiển thị kết quả tìm kiếm.

Từ những khảo sát trên chúng ta đưa ra cấu trúc chung cho đường dẫn như sau :

```
https://www.freeimages.com/search/{name}/{num_page}
```

Sau khi biết được cấu trúc đường dẫn chúng ta phải tìm cách lấy đường dẫn ảnh hiển thị trên trang web. Trong trang kết quả, chúng ta xem nguồn trang bằng cách nhấn tổ hợp phím Ctr+U (đây là tổ hợp phím dành cho google chrome) hoặc chuột phải + xem nguồn trang.

```
< > C ⓘ view-source:https://www.freeimages.com/search/cat/2
Tự ngắt dòng □
1
2
3
4
5 <!DOCTYPE html>
6 <html lang="en">
7
8 <head>
9   <meta charset="UTF-8">
10  <meta http-equiv="X-UA-Compatible" content="IE=edge">
11  <meta name="viewport" content="width=device-width, initial-scale=1.0">
12
13  <link rel="alternate" hreflang="en" href="https://www.freeimages.com/search/cat/2" />
14  <link rel="alternate" hreflang="de" href="https://www.freeimages.com/de/search/cat/2" />
15  <link rel="alternate" hreflang="es" href="https://www.freeimages.com/es/search/cat/2" />
16  <link rel="alternate" hreflang="fr" href="https://www.freeimages.com/fr/search/cat/2" />
17  <link rel="alternate" hreflang="it" href="https://www.freeimages.com/it/search/cat/2" />
18  <link rel="alternate" hreflang="ja" href="https://www.freeimages.com/jp/search/cat/2" />
19  <link rel="alternate" hreflang="ko" href="https://www.freeimages.com/kr/search/cat/2" />
20  <link rel="alternate" hreflang="nl" href="https://www.freeimages.com/nl/search/cat/2" />
21  <link rel="alternate" hreflang="pl" href="https://www.freeimages.com/pl/search/cat/2" />
22  <link rel="alternate" hreflang="pt" href="https://www.freeimages.com/pt/search/cat/2" />
23  <link rel="alternate" hreflang="pt-br" href="https://www.freeimages.com/pt-br/search/cat/2" />
24  <link rel="alternate" hreflang="ru" href="https://www.freeimages.com/ru/search/cat/2" />
25  <link rel="alternate" hreflang="sv" href="https://www.freeimages.com/se/search/cat/2" />
26  <link rel="alternate" hreflang="tr" href="https://www.freeimages.com/tr/search/cat/2" />
27  <link rel="alternate" hreflang="cn" href="https://www.freeimages.com/cn/search/cat/2" />
28  <link rel="alternate" hreflang="tw" href="https://www.freeimages.com/tw/search/cat/2" />
29  <link rel="alternate" hreflang="x-default" href="https://www.freeimages.com/search/cat/2" />
```

Hình 6: Mã nguồn của trang web

Kết quả hiển thị một văn bản gồm các dòng mã viết bằng ngôn ngữ html, trong văn bản này, chứa toàn bộ thông tin trang mà bạn đang truy cập bao gồm cả đường dẫn của các hình ảnh. Chúng ta nhận

thấy, các đường dẫn hình ảnh đều trong các khối mã có cấu trúc như hình.

```

</div>
<div class="grid-image-wrapper w-full h-full">

</div>

```

Hình 7: Khối mã chứa đường dẫn ảnh

Lưu ý: Mình có khảo sát thì thấy đa phần đường dẫn ảnh nằm trong khái code trên, tuy nhiên có một số cấu trúc khác cần phải xử lý thêm, phần này các bạn tìm hiểu thêm nhé!

Bây giờ việc crawl dữ liệu chỉ đơn giản là bài toán xử lý string thôi sao! Thật may là chúng ta có rất nhiều thư viện để hỗ trợ chúng ta làm điều này dễ dàng hơn đó là sử dụng BeautifulSoup, ....

Vậy làm thế nào chúng ta có thể lập trình để truy cập vào trang web và lấy đường dẫn hình ảnh từ trang web đó. Chúng ta hãy đến với khái niệm Web crawling.

Web crawling là quá trình tự động trích xuất các thông tin từ các trang web và lưu trữ nó dưới một định dạng phù hợp. Chương trình mà thực hiện công việc này gọi là web crawler.

Và để thực hiện thu thập hình ảnh từ freeimages.com chúng ta tiến hành lập trình theo hai bước sau:

- **Get urls to txts: Gửi request đến trang freeimages.com, lưu các đường dẫn của ảnh vào file txt theo từng topic.**



Hình 8: Get urls to txts

Thư viện BeautifulSoup là một thư viện của Python cho phép chúng ta lấy dữ liệu từ HTML đơn giản và hiệu quả. Để xử lý thông tin nhanh hơn, chúng ta sử dụng thư viện threading để xử lý đa

luồng. Để phân tích cú pháp url sử dụng thư viện urllib.parse. Sử dụng thư viện time để xử lý một số tác vụ liên quan đến thời gian và sử dụng thư viện os để xử lý các thao tác với file, thư mục...

```
#Crawler.ipynb
from bs4 import BeautifulSoup
from urllib.parse import urljoin, urlparse
import urllib.request
import time
from threading import Thread
import os
```

Để lấy được đường dẫn hình ảnh trên một trang, chúng ta xây dựng class GetImagesfromPage. Chúng ta sẽ khai báo class và tạo hàm khởi tạo với các thuộc tính nThreads: Số luồng hoạt động, npage: Số lượng trang mà mình sẽ lấy ảnh, url\_page: Đường dẫn đến trang kết quả, result\_url: Một list lưu lại các đường dẫn ảnh get được.

```
#class GetImagesfromPages()
class GetImagesfromPages():
    def __init__(self, nThreads, npage, url_page):
        self.nThreads = nThreads
        self.npage = npage
        self.url_page = url_page
        self.result_urls = []
```

Để kiểm tra một đường dẫn có hợp lệ không, chúng ta tạo hàm is\_valid()

```
#class GetImagesfromPages()
def is_valid(self, url):
    """
    Checks whether 'url' is a valid URL.
    """
    parsed = urlparse(url)
    return bool(parsed.netloc) and bool(parsed.scheme)
```

Theo khảo sát trên trang freeimages.com thì đường dẫn của hầu hết hình ảnh đều nằm trong khái code sau.

```
</div>
<div class="grid-image-wrapper w-full h-full">

</div>
```

Hình 9: Đường dẫn ảnh trong trang web

Chúng ta sẽ xây dựng hàm get\_all\_images() thực hiện get html từ đường dẫn, sau đó phân tích

dữ liệu để lấy được đường dẫn của toàn bộ hình ảnh từ url.

Lưu ý: Ở hàm này mình lấy đường dẫn ảnh theo "src", tuy nhiên hình ảnh này chất lượng thấp, để lấy được hình ảnh chất lượng cao thì bạn lấy ảnh theo "data-src". Các bạn khi dùng data-src cần thêm các bước xử lý chuyên sâu vì vậy mình sẽ không trình bày trong bài hướng dẫn này!

```
#class GetImagesfromPages()
def get_all_images(self, url):
    """
    Returns all image URLs on a single 'url'
    """
    soup = BeautifulSoup(urllib.request.urlopen(url), "html.parser")
    urls = []
    for img in soup.find_all("img"):
        img_url = img.attrs.get("src")

        if not img_url:
            # if img does not contain src attribute, just skip
            continue

        # make the URL absolute by joining domain with the URL that is just
        # extracted
        img_url = urljoin(url, img_url)
        # remove URLs like '/hsts-pixel.gif?c=3.2.5'
        try:
            pos = img_url.index("?")
            img_url = img_url[:pos]
        except ValueError:
            pass

        # finally, if the url is valid
        if self.is_valid(img_url):
            urls.append(img_url)

    return urls
```

Để lấy danh sách tất cả link ảnh để dùng download ở bước sau, chúng ta xây dựng hàm main()

```
#class GetImagesfromPages()
def main(self, start, end):
    for i in range(start, end):
        try:
            self.result_urls.extend(self.get_all_images(self.url_page + str(i)))
        except:
            pass
```

Cuối cùng, chúng ta xây dựng hàm `__call__()` để xử lí đa luồng, method call sẽ dùng nhiều thread để cùng xử lý các url và mỗi thread dùng hàm `main()` lấy danh sách các đường link ảnh. Sau đó thì sẽ tập hợp các list của từng thread lại thành 1 list các link ảnh

```
#class GetImagesfromPages():
    def __call__(self):
        # Create Threads
        threads = []

        batch = self.npage//self.nThreads
        for i in range(0, self.npage, batch):
            start = i
            end = i + batch

            if end >= self.npage:
                end = self.npage + 1

            threads.append(Thread(target=self.main, args = (start, end)))

        start = time.time()
        for i in range(self.nThreads):
            threads[i].start()
        for i in range(self.nThreads):
            threads[i].join()
        end = time.time()

        print(f"Time handle pages = {end - start:.2f}s", )

        return self.result_urls
```

Sau khi xây dựng class `GetImagesfromPages`, chúng ta tạo hàm `urls_to_txt()`. Hàm này thực hiện ghi lại đường dẫn vào file txt với đầu vào là `topic_names`, `topics`, `urltopic`, `n_page`

```
def urls_to_txts(topic_names, topics, urltopic, n_page, n_threads):
    for dir, names in zip(topic_names, topics):
        dir_path_urls = f"data/{dir}/urls"
        if not os.path.exists(dir_path_urls):
            os.makedirs(dir_path_urls)

        for name in names:
            result_of_name = []
            for key in urltopic.keys():
                res = GetImagesfromPages(min(n_threads, n_page//2), n_page,
                                         urltopic[key].format(name = name))()

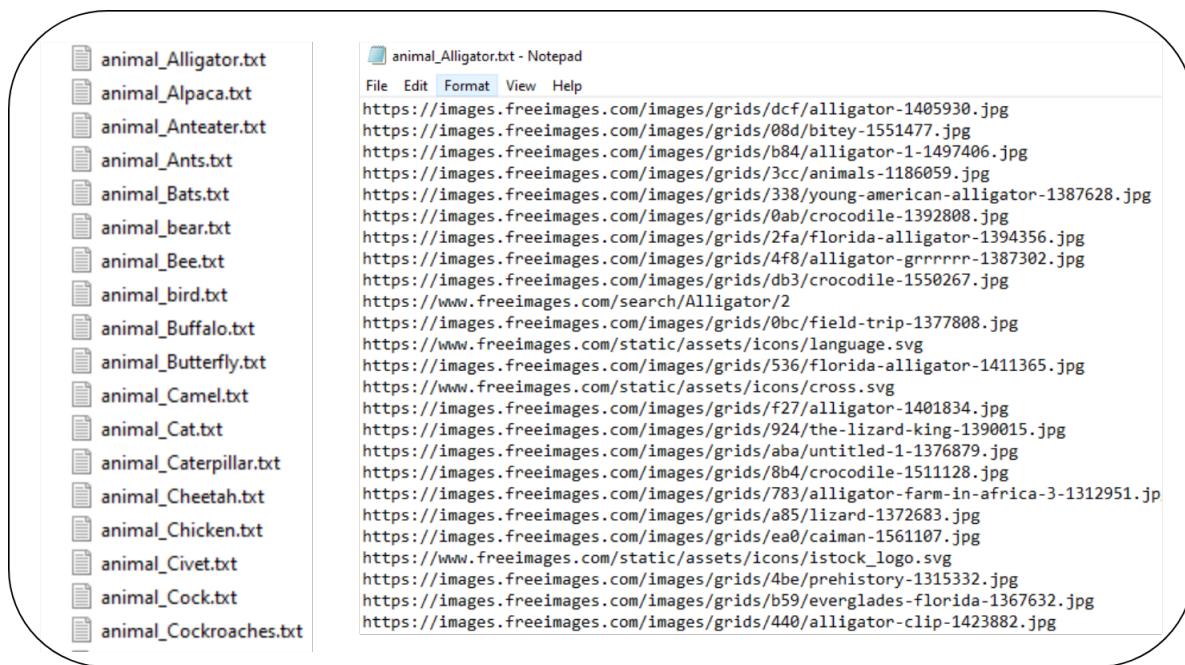
                if len(res) > 0:
                    res = list(set(res))
                    result_of_name.extend(res)

            print(f"{dir_path_urls}/{dir}_{name}.txt have {len(result_of_name)}")
            images \n)
            strResult = '\n'.join(result_of_name)
            with open(f"{dir_path_urls}/{dir}_{name}.txt", "w") as f:
                f.write(strResult)
```

Cuối cùng chúng ta chạy chương trình và xem kết quả thu được

```
urltopic = {
    "freeimages": "https://www.freeimages.com/search/{name}/"
}

topic_names = ["animal", "plant", "furniture", "scenery"]
topics = [animal, plant, furniture, scenery]
n_threads = 3
n_page = 6
urls_to_txts(topic_names=topic_names, topics=topics, urltopic=urltopic,
n_page=n_page, n_threads=n_threads)
```



Hình 10: Thư mục file đường dẫn ảnh

- Tải các hình ảnh theo từng topic vào các thư mục với tên tương ứng với từng topic

Sau khi có đường dẫn các hình ảnh, chúng ta tải các hình ảnh bằng cách sử dụng thư viện `urllib.request`.

Đầu tiên chúng ta khai báo các thư viện cần thiết và xây dựng class `DownloadImagesFromUrls()`.

```
import urllib.request
from threading import Thread
import time
import requests
import random
import os

class DownloadImagesFromUrls():

    def __init__(self, urls):
        self.urls = urls
        self.image_paths = []
        self.threads = []

    def download_images(self):
        for url in self.urls:
            thread = Thread(target=self.download_image, args=(url,))
            self.threads.append(thread)
            thread.start()

    def download_image(self, url):
        response = requests.get(url)
        if response.status_code == 200:
            image_path = os.path.join('images', url.split('/')[-1])
            with open(image_path, 'wb') as f:
                f.write(response.content)
            self.image_paths.append(image_path)
            print(f"Downloaded {image_path}")

    def get_image_paths(self):
        return self.image_paths
```

```

def __init__(self, nThreads, urls, destinate_folder):
    self.nThreads = nThreads
    self.urls = urls
    self.n = len(urls)
    self.destinate_folder = destinate_folder
# Func target
def download_url(self, start, end):

    for i in range(start, end):
        a = random.random()
        try:
            urllib.request.urlretrieve(self.urls[i],
f"{self.destinate_folder}/{a}.jpg")
        except:
            print(f"cannot access {self.urls[i]}")
            print('.', end=" ")

def __call__(self):

    threads = []
    batch = self.n//self.nThreads
    for i in range(0, self.n, batch):
        start = i
        end = i + batch

        if end >= self.n:
            end = self.n

        threads.append(Thread(target=self.download_url, args = (start,
end)))

    start = time.time()
    for i in range(self.nThreads):
        threads[i].start()
    for i in range(self.nThreads):
        threads[i].join()
    end = time.time()

    print(f"\nTime handle download urls = {end - start:.2f}s\n", )

```

Tiếp theo chúng ta xây dựng hàm get\_image\_from\_txts().

```

def get_image_from_txts(topic_names, topics):
    for dir, names in zip(topic_names, topics):

        dir_path_images = f"images"
        dir_path_urls = f"data/{dir}/urls"
        if not os.path.exists(dir_path_images):
            os.makedirs(dir_path_images)

        txts = [name for name in os.listdir(dir_path_urls) if
name.endswith(".txt")]

```

```

for txt in txts:
    folder_txt = f"{dir_path_urls}/{txt}"
    with open(folder_txt, "r") as f:
        content_txt = f.readlines()

    folder_image = f"{dir_path_images}/{txt}"
    if not os.path.exists(folder_image[:-4]):
        os.makedirs(folder_image[:-4])
    print(folder_image[:-4])

n_threads = 10
DownloadImagesFromUrls(min(n_threads, len(content_txt)//2),
content_txt, folder_image[:-4])()

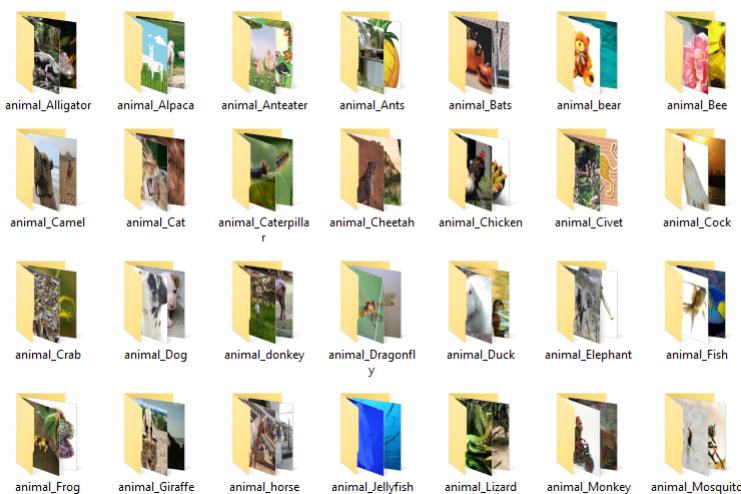
```

Chúng ta chạy chương trình và thu được kết quả là các thư mục ảnh theo từng topic.

```

topic_names = ["animal", "plant", "furniture", "scenery"]
topics = [animal, plant, furniture, scenery]
get_image_from_txts(topic_names=topic_names, topics=topics)

```

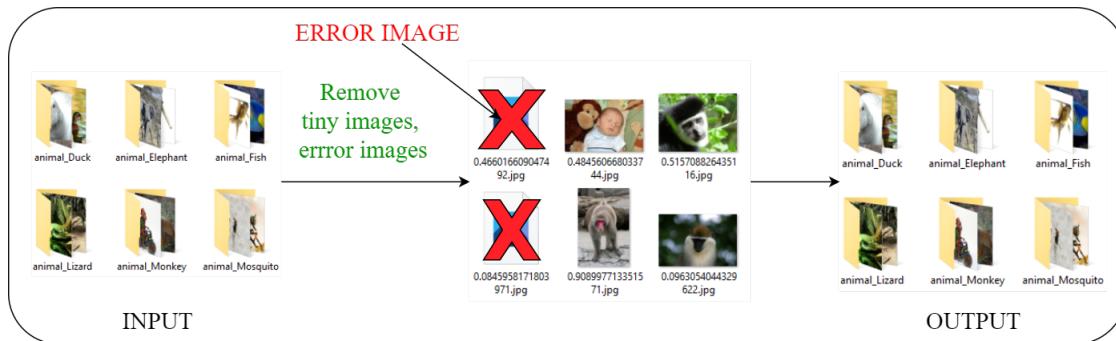


Hình 11: Thư mục ảnh thu được sau khi crawl từ freeimages.com

==> Về cơ bản, chúng ta đã phân tích các vấn đề về crawl dữ liệu, đưa ra các giải pháp sử dụng thư viện có sẵn để xây dựng công cụ thu thập dữ liệu hình ảnh trên freeimg.com. Bạn có thể phát triển công cụ này tốt hơn nữa bằng các kỹ thuật xử lý đa luồng.....code chi tiết phần này bạn tham khảo tại đây: [Crawler.ipynb](#)

### 2.1.2 Xử lí dữ liệu-Data preprocessing

Trong thực tế sau khi thu thập dữ liệu về có thể xuất hiện các vấn đề như dữ liệu bị lỗi, ví dụ: hình ảnh quá mờ không chứa thông tin, kích thước hình ảnh quá nhỏ, hình ảnh bị hỏng, ... Vì vậy xử lí dữ liệu là bước tiền đề quan trọng để có được bộ dữ liệu tốt.



Hình 12: Data preprocessing

Đầu tiên chúng ta sẽ thêm các thư viện cần thiết.

```
import os
from PIL import Image
import numpy as np
import warnings
warnings.filterwarnings("ignore", "(Possibly )?corrupt EXIF data", UserWarning)
Image.MAX_IMAGE_PIXELS = None
```

Tiếp theo chúng ta sẽ xây dựng hàm để xử lý ảnh:

- Loại bỏ những hình ảnh lỗi (Không đọc được ảnh,...)
- Loại bỏ những hình ảnh có kích thước nhỏ hơn 10
- Loại bỏ hình ảnh có channel khác 3 (chỉ nhận ảnh màu RGB). Chúng ta đặt một chuẩn chung của hình ảnh để có thể tính toán

Lưu ý: Các điều kiện ở trên mình chọn theo bộ dữ liệu của mình, với những bộ dữ liệu khác bạn có thể chọn các thông số khác sao cho phù hợp.

```
def processing_data(images_path):
    dic_categories = {'animal' : [], 'plant' : [], 'furniture' : [], 'scenery' : []}
    count = 0

    for folder in os.listdir(images_path):
        if folder.split("_")[0] in dic_categories:
            path = images_path + folder
            list_dir = [path + '/' + name for name in os.listdir(path) if
            name.endswith((".jpg", ".png", ".jpeg"))]
            for p in list_dir:
                try:
                    # Open image
                    img = Image.open(p) # open the image file

                    # Verify image
                    img.verify() # verify that it is, in fact an image

                    # Open image
                    img = Image.open(p) # open the image file
                    count += 1
                except:
                    pass
```

```

# Check width of image
if img.width < 10:
    print("Image too small: ", p)
    os.remove(p)

# Only 3 channle image (color image)
img = np.asarray(img)
if img.shape[2] != 3:
    os.remove(p)

except Exception as e:
    print(e)
    count += 1
    print("error: ", p)
    os.remove(p)

```

Code chi tiết phần này, bạn xem ở [PreprocessingData.ipynb](#)

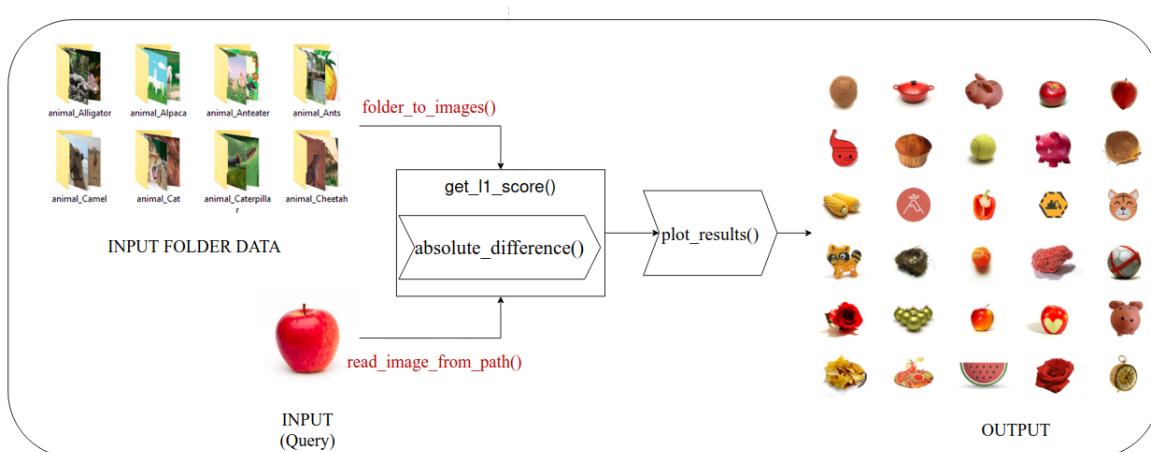
## 2.2 Xây dựng chương trình truy vấn ảnh

Để có thể tìm được các hình ảnh có liên quan đến hình ảnh truy vấn, ta có thể sử dụng các công thức được dùng để đo sự tương đồng giữa hai ảnh.

### 2.2.1 Tính độ tương đồng với độ đo L1

Chúng ta xây dựng một hàm *absolute\_difference()* tính độ tương đồng giữa các hình ảnh. Trong ví dụ này chúng ta sẽ sử dụng hàm L1. Hàm L1 có công thức tính như sau:

$$l1\_norm(\vec{a}, \vec{b}) = \sum_{i=1}^N |a - b|$$



Hình 13: Image Retrieval với L1

Đầu tiên chúng ta sẽ import một số thư viện cần thiết. Để đọc ảnh chúng ta sử dụng thư viện PIL, để xử lý ma trận chúng ta sử dụng numpy, để thao tác với thư mục, file chúng ta sử dụng thư viện os, sử dụng matplotlib để hiển thị kết quả.

---

```
from PIL import Image # Doc anh
import numpy as np # Xu ly ma tran
import os # Thao tac lay file, move file cua OS
import matplotlib.pyplot as plt
```

---

Sau đó để thực hiện tính toán trên các hình ảnh, chúng ta sẽ đọc ảnh, resize về kích thước chung (thì mới áp dụng được các phép đo) và chuyển đổi nó về dạng numpy:

```
def read_image_from_path(path, size):
    im = Image.open(path).resize(size)
    return np.asarray(im, dtype=np.float32)

def folder_to_images(folder, size):

    list_dir = [folder + '/' + name for name in os.listdir(folder) if
name.endswith((".jpg", ".png", ".jpeg"))]

    i = 0
    images_np = np.zeros(shape=(len(list_dir), *size, 3))
    images_path = []
    for path in list_dir:
        try:
            images_np[i] = read_image_from_path(path, size)
            images_path.append(path)
            i += 1

        except Exception:
            print("error: ", path)
#            os.remove(path)

    images_path = np.array(images_path)
    return images_np, images_path
```

---

Tiếp theo để tính toán độ tương đồng giữa các hình ảnh chúng ta sẽ tạo hàm sử dụng phép đo L1 *absolute\_difference()*:

```
def absolute_difference(query, X):
    axis_batch_size = tuple(range(1, len(X.shape)))
    return np.sum(np.abs(X - query), axis=axis_batch_size)
```

---

Đến đây chúng ta sẽ thực hiện tính toán để tính độ tương đồng giữa ảnh input và các hình ảnh trong bộ dữ liệu. Chúng ta sẽ tạo hàm *get\_l1\_score()*, hàm này sẽ trả về danh sách hình ảnh và giá trị độ tương đồng với từng ảnh.

```
def get_l1_score(root_img_path, query_path, size):
    dic_categories = ['scenery', 'furniture', 'animal', 'plant']
    query = read_image_from_path(query_path, size)
    ls_path_score = []
    for folder in os.listdir(root_img_path):
        if folder.split('_')[0] in dic_categories:
            path = root_img_path + folder
```

---

```

# mang numpy nhieu anh, paths
images_np, images_path = folder_to_images(path, size)
rates = absolute_difference(query, images_np)
ls_path_score.extend(list(zip(images_path, rates)))
return query, ls_path_score

```

Chúng ta sẽ tạo hàm để hiển thị kết quả, kết quả trả về là 30 hình ảnh có độ tương đồng với hình ảnh Input cao nhất. Đối với mỗi độ đo khác nhau mà độ tương đồng cao nhất có thể có giá trị lớn nhất hoặc nhỏ nhất. Trong ví dụ này chúng ta sử dụng độ đo L1, vì vậy độ tương đồng cao nhất sẽ có giá trị nhỏ nhất.

```

def plot_results(query, ls_path_score):
    # Show query image
    plt.imshow(query/255.0)
    # Score
    fig = plt.figure(figsize=(15, 15))
    columns = 5
    rows = 6
    for i, path in enumerate(sorted(ls_path_score, key=lambda x : x[1])[:30], 1):
        img = np.random.randint(10, size=(10,10))
        fig.add_subplot(rows, columns, i)
        plt.imshow(plt.imread(path[0]))
        plt.axis("off")
    plt.show()

```

Tiếp theo chúng ta sẽ truy vấn hình ảnh với 3 độ đo L2, Cosine Similarity, Correlation Coefficient. Chúng ta sẽ thực hiện code tương tự như độ đo L1.

### 2.2.2 Truy vấn hình ảnh với độ đo L2

$$l2\_norm(\vec{a}, \vec{b}) = \sqrt{\sum_{i=1}^N (a_i - b_i)^2}$$

Chúng ta tạo hàm tính độ tương đồng L2 mean\_square\_difference()

```

def mean_square_difference(query, X):
    axis_batch_size = tuple(range(1, len(X.shape)))
    return np.mean((X - query)**2, axis=axis_batch_size)

```

Chúng ta tạo hàm get\_l2\_score(), hàm này tương tự hàm get\_l1\_score(), chỉ cần thay đổi tại biến rates.

```

# rates = absolute_difference(query, images_np)
rates = mean_square_difference(query, images_np)

```

Vì với độ đo L2 này thì giá trị càng nhỏ sẽ càng giống nhau. Nên để hiển thị kết quả chúng ta sử dụng hàm plot\_results() như phép đo L1.

### 2.2.3 Truy vấn hình ảnh với độ đo Cosine Similarity

$$\text{cosine\_similarity}(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \|\vec{b}\|} = \frac{\sum_{i=1}^N a_i b_i}{\sqrt{\sum_{i=1}^N a_i^2} \sqrt{\sum_{i=1}^N b_i^2}}$$

Chúng ta tạo hàm tính độ tương đồng cosine\_similarity()

```
def cosine_similarity(query, X):
    axis_batch_size = tuple(range(1, len(X.shape)))
    # Sử dụng norm
    query_norm = np.sqrt(np.sum(query**2))
    X_norm = np.sqrt(np.sum(X**2, axis=axis_batch_size))
    return np.sum(X * query, axis=axis_batch_size) / (query_norm*X_norm +
np.finfo(float).eps)
```

Chúng ta tạo hàm get\_cosine\_similarity\_score(), hàm này tương tự hàm get\_l1\_score(), chỉ cần thay đổi tại biến rates.

```
# def get_cosine_similarity_score(query, X)
rates = cosine_similarity(query, images_np)
```

Để hiển thị kết quả chúng ta sử dụng hàm plot\_results(), tuy nhiên ở hàm này chúng ta sẽ sắp xếp giá trị giảm dần từ lớn đến nhỏ vì với độ đo này thì giá trị càng lớn sẽ càng giống nhau, cho nên trong hàm sorted() chúng ta sử dụng reverse = True.

```
# def plot_results()
sorted(ls_path_score, key=lambda x : x[1], reverse = True)
```

### 2.2.4 Truy vấn hình ảnh với độ đo Correlation Coefficient

$$p_{xy} = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_x \sigma_y} = \frac{N(\Sigma_i x_i y_i) - (\Sigma_i x_i)(\Sigma_i y_i)}{\sqrt{N \sum_i x_i^2 - (\Sigma_i x_i)^2} \sqrt{N \sum_i y_i^2 - (\Sigma_i y_i)^2}}$$

Chúng ta tạo hàm tính độ tương đồng correlation\_coefficient()

```
def correlation_coefficient(query, X):
    axis_batch_size = tuple(range(1, len(X.shape)))

    # Sử dụng cua mean
    query_mean = query - np.mean(query)
    X_mean = X - np.mean(X, axis=axis_batch_size, keepdims=True)

    # Sử dụng cua norm
    query_norm = np.sqrt(np.sum(query_mean**2))
    X_norm = np.sqrt(np.sum(X_mean**2, axis=axis_batch_size))

    return np.sum(X_mean * query_mean, axis=axis_batch_size) / (query_norm*X_norm +
np.finfo(float).eps)
```

Chúng ta tạo hàm `get_correlation_coefficient_score()`, hàm này tương tự hàm `get_ll_score()`, chỉ cần thay đổi tại biến `rates`.

```
# def correlation_coefficient(query, X)
rates = correlation_coefficient(query, images_np)
```

Vì với độ đo Correlation Coefficient này thì giá trị càng lớn sẽ càng giống nhau. Nên để hiển thị kết quả chúng ta sử dụng hàm `plot_results()` như độ đo cosine similarity.

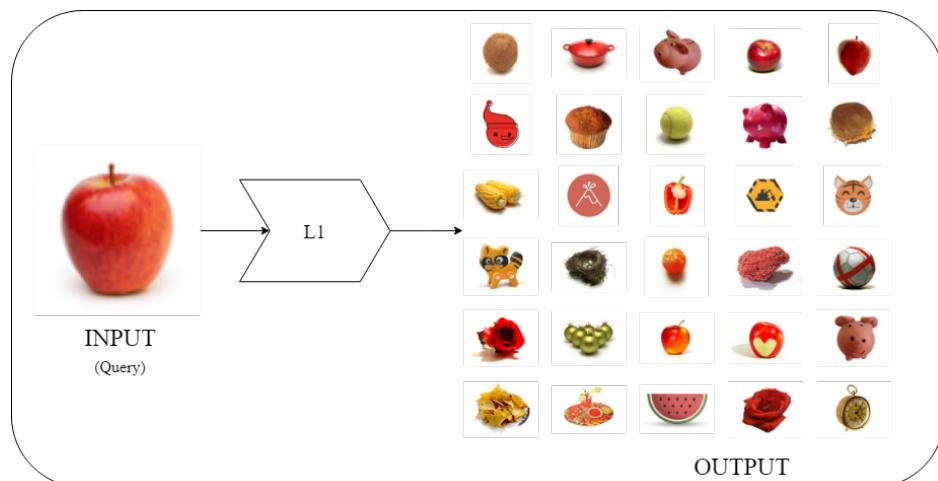
Code chi tiết phần này, bạn xem ở [ExperimentMethods.ipynb](#)

### 3 Thủ nghiêm và đánh giá kết quả

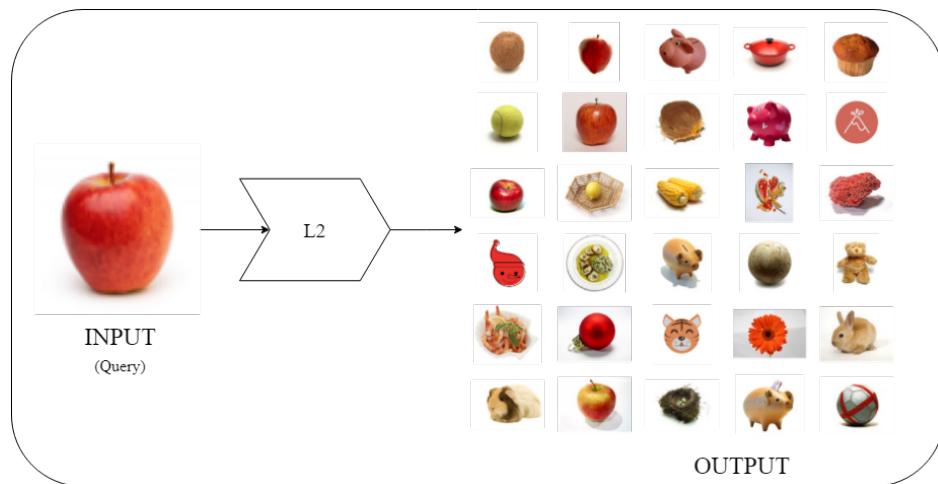
Cuối cùng là phần thú vi nhất, hãy xem thành quả mà chúng ta đạt được:

```
root_img_path = "/content/images/"
query_path = "apple.jpg"
size = (80, 80)
query, ls_path_score = get_l1_score(root_img_path, query_path, size)
#query, ls_path_score = get_l2_score(root_img_path, query_path, size)
#query, ls_path_score = get_Cosine_Similarity_score(root_img_path, query_path, size)
#query, ls_path_score = get_Correlation_Coefficient_score(root_img_path, query_path,
size)
plot_results(query, ls_path_score)
```

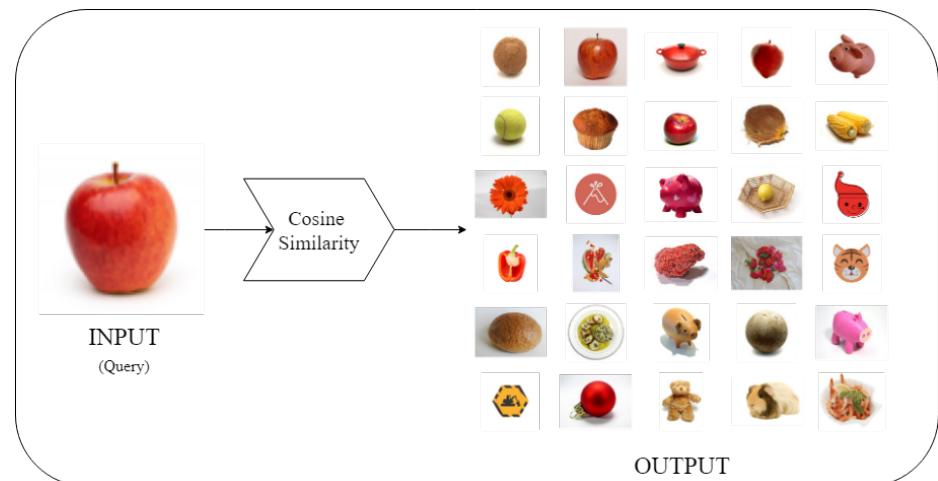
Và đây là kết quả chúng ta đạt được, nhìn chung khá thú vị:



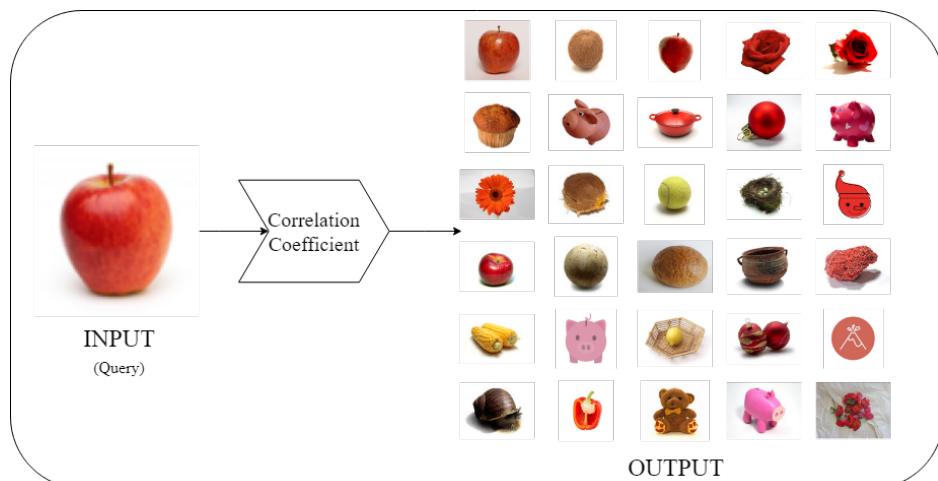
Hình 14: Kết quả sử dụng Image Retrieval với L1



Hình 15: Kết quả sử dụng Image Retrieval với L 2



Hình 16: Kết quả sử dụng Image Retrieval với Cosine Similarity



Hình 17: Kết quả sử dụng Image Retrieval với Correlation Coefficient

Qua kết quả thu được chúng ta có thể thấy rằng phương pháp truy vấn ảnh sử dụng phép đo Correlation Coefficient cho kết quả tốt nhất, cosine similarity cho kết quả khá, kế sau đó là L2 và L1.

Nhìn chung việc chỉ sử dụng các phép đo cơ bản đã cho kết quả khá ấn tượng, tuy nhiên kết quả này chưa đủ tốt để sử dụng trong thực tế.

## 4 Kết Luận

Như vậy, trong project này chúng ta đã cùng nhau tìm hiểu về bài toán images retrieval và thu được một số kiến thức sau:

- Tìm hiểu về bài toán images retrieval
- Nắm được các bước xây dựng một chương trình cho bài toán thực tế.
- Nắm được một số kỹ thuật để thu thập dữ liệu, cách phân tích, giải quyết vấn đề cơ bản.
- Hiểu thêm về các độ đo và ứng dụng của chúng
- Nâng cao kỹ năng lập trình...

Tổng hợp file solution: [ImageRetrievalSolution](#)

## 5 Phản hồi từ bạn đọc

Nếu bạn đọc thấy chỗ nào có thể cải thiện để tải liệu này đúng hơn, dễ đọc hơn, hay đơn giản là trình bày rõ ràng hơn; thì inbox thông tin cho nhóm qua fan page AI VIETNAM. Xin cảm ơn các bạn trước.

- *Hết* -