



Gestión avanzada de datos

Proyecto

Integrantes

- Hoet, Leonardo Alfonso
- Klug, Ezequiel Federico
- Ponsiano, María Victoria

Docentes

- Ing. Andres Pascal
- Ing. Adrián Planas

Fecha de entrega: 15/12/2021

Índice

Índice	2
Introducción	3
Desarrollo	4
Fuente de datos	4
Extracción de características	5
Estructura de la base de datos	5
Inserción de las imágenes en la base de datos.	6
Procedimientos utilizados	6
Distancia coseno	6
Selección incremental de pivotes	7
FHQT+	7
Histograma	7
Consulta de objetos similares	7
Creación de la aplicación	9
Interfaz de usuario	9
Resultados	10

Introducción

El siguiente trabajo tiene como objetivo lograr que los alumnos desarrollen una aplicación que permite buscar objetos por Similitud. La aplicación desarrollada por el grupo permite la búsqueda por similitud de imágenes del espacio.

Desarrollo

A continuación, se detalla la fuente de datos utilizada, cómo se obtuvieron las imágenes, cómo se extrajeron características de las imágenes, las procedures utilizados y la creación de la aplicación para luego poder realizar la búsqueda por similitud.

Fuente de datos

El dataset elegido está compuesto por imágenes del cielo nocturno capturadas por el proyecto DSS (Digital Sky Survey). El mismo busca obtener una base de datos de una gran parte de la bóveda celeste. Estos datos se encuentran disponibles para el público general. Entre los mismos podemos encontrar diferentes tipos como espectrometrías, datos en formato FITS e imágenes en JPG. En este trabajo usaremos las imágenes en JPG por una cuestión de simplicidad.

Los datos fueron obtenidos de la versión DR12. Estos se pueden ver en el siguiente [link](#). Se descargaron 5076 imágenes utilizando el siguiente script para poder realizar la descarga de forma automatizada:

```
import urllib.request as ur
from pyquery import PyQuery as pq
import os, ssl
if (not os.environ.get('PYTHONHTTPSVERIFY', '')) and
    getattr(ssl, '_create_unverified_context', None)):
    ssl._create_default_https_context = ssl._create_unverified_context

base_url = "https://dr12.sdss.org"

def query_url(ra: float, dec: float) -> str:
    return base_url + f"/fields/raDec?ra={ra}&dec={dec}"

def img_url(path: str) -> str:
    return base_url + path

# Downloads an image of the celestial sphere in the Ra,Dec pos
# out_name is the output name of the file
def download_image(ra: float, dec: float, out_name: str):
    try:
        print(f"ra: {ra}, dec: {dec}")
        response = ur.urlopen(query_url(ra,dec))
    except ur.HTTPError as e:
        print('HTTPError: {}'.format(e.code))
        return dec+25
    else:
        d = pq(query_url(ra, dec))
```

```

    path = d('#jpg').attr("src")
    if (path):
        print(f"Downloading imagen{ra}_{dec}.jpg")
        resource = ur.urlopen(img_url(path))
        with open(out_name, "wb") as f:
            f.write(resource.read())
        return dec+0.2
    else:
        print(f"imagen{ra}_{dec}.jpg Doesn't Exist")
        return dec+25

count=0
i=0
while i < 40:
    j=0.2
    i+=0.2
    while j<40:
        count+=1
        j=download_image(i, j, f"imagen{i}_{j}.jpg")

```

Extracción de características

Una vez obtenidas las 5076 imágenes, se continúa con la extracción de características de cada imagen para poder así, guardar los vectores que representan las imágenes en la base de datos.

Para lograr obtener las características, se utilizó la librería de python llamada 'img2vec_pytorch'¹, que está entrenado por redes neuronales convolucionales. La librería ofrece distintos modelos, el utilizado por nuestro grupo es el Resnet-18.

Utilizando el siguiente script, se realiza la iteración sobre las imágenes previamente descargadas y se extraen los vectores característicos. Un ejemplo de uso de la librería es el siguiente:

```

from img2vec_pytorch import Img2Vec
from PIL import Image

# Initialize Img2Vec with GPU
img2vec = Img2Vec(cuda=True)

# Read in an image (rgb format)
img = Image.open('img1.jpg')

```

¹ <https://github.com/christiansafka/img2vec>

```
# Get a vector from img2vec, returned as a torch FloatTensor
vec = img2vec.get_vec(img, tensor=True)
```

El script que nosotros utilizamos para poder extraer las características de las imágenes se llama `img_2_vec.py`.

Estructura de la base de datos

```
CREATE TABLE public.imagenes (
    id integer NOT NULL,
    path character varying(255),
    vector double precision[],
    id_hoja bigint,
    web_path character varying(2083)
);

CREATE TABLE histograma(intervalo numrange, nivel integer);

CREATE TABLE public.pivote (
    id_pivote bigint NOT NULL,
    nivel bigint NOT NULL,
    vector double precision[]
);

CREATE TABLE public.tree (
    id_nodo integer NOT NULL,
    id_padre bigint,
    nivel bigint,
    intervalo_max numrange,
    intervalo_act numrange
);
```

En la tabla imágenes almacenamos los vectores que representan a las imágenes junto con un id, el path local de la máquina donde se encuentra guardada la imagen, el id de la hoja donde se encuentra en el árbol, y la url que permite visualizarla y descargarla desde la web.

La tabla histograma contiene todos los intervalos máximos que se calculan con el SP `histograma()`.

La tabla pivote contiene los id de los pivotes con su nivel y el vector que representa la imagen. Estos pivotes son elegidos con el algoritmo de selección incremental.

La tabla tree contiene el id del nodo, el id del padre, el nivel, el intervalo máximo que se utiliza para insertar y el intervalo actual que se utiliza para realizar las búsquedas.

Inserción de las imágenes en la base de datos.

Para poder insertar los vectores característicos en la base de datos se utilizó el script `connect.py` que se conecta a la base de datos utilizando el script `config.py` y el archivo `database.ini` e inserta todos los objetos leídos de un archivo `data.txt` en formato json que contiene los vectores característicos de las 5076 imágenes.

Procedimientos utilizados

Distancia coseno

Para el cálculo de distancias entre vectores, se creó una función que calcula la distancia coseno entre 2 vectores. Cabe aclarar que puede tomar valores entre 0 y 1. En caso de que la distancia sea 1, significa que los vectores son muy distintos y si da 0, los vectores son iguales.

El archivo completo se llama `coseno.sql`.

```
CREATE OR REPLACE FUNCTION public.cosine_distance(IN vector1 double
precision[], IN vector2 double precision[])
    RETURNS double precision
    LANGUAGE 'plpgsql'
AS $BODY$
BEGIN
    RETURN ((SELECT 1 - ((select public.dot_product(vector1, vector2)
as dot_pod)/((select public.vector_norm(vector1) as norm1) * (select
public.vector_norm(vector2) as norm2))) AS cosine_distance));
END; $BODY$;
```

Selección incremental de pivotes

Si bien este procedimiento fue realizado durante la materia, hubo que modificar la función de distancia que utilizaba. En lugar de utilizar la distancia de levenshtein, ahora se utiliza la distancia coseno.

Se optó por utilizar dicho método ya que obtiene el mejor conjunto de pivotes en la mayoría de los casos.

El archivo se denomina `seleccion_incremental.sql`

FHQT+

Dicho procedimiento también fue realizado durante la materia, pero hubo que adaptarlo para que funcione con distancias continuas y utilice la distancia coseno en lugar de la distancia de levenshtein. Por lo tanto, en lugar de buscar distancias por igualdad, ya que se trabajaba con distancias discretas, ahora se trabaja con dos rangos, máximo y actual. El primero utilizado para determinar en que rama del árbol agregar un nuevo elemento y el segundo para realizar la búsqueda de un elemento en el árbol.

Se cuenta con 3 archivos denominados `fhq_insert.sql`, `fhq_update.sql` y `fhq_delete.sql`. Estas funciones son llamadas mediante triggers que se disparan cuando se realiza una inserción, modificación o eliminación de la tabla 'imagenes', para actualizar según corresponda la tabla tree.

Histograma

Como mencionamos anteriormente, el FHQT+ utiliza rangos de distancias. Para poder dividir el espacio en rangos, se creó una función denominada histograma con la cual se obtiene una muestra de todas las distancias entre las imágenes de la base de datos de tal manera de conocer cómo están distribuidos los datos y poder armar los intervalos máximos.

El archivo se denomina `seleccion_incremental.sql`

Consulta de objetos similares

Para poder realizar una consulta por objetos similares en la base de datos, se creó una función que acepta como parámetro un vector, el cual representa la imagen, y un radio de búsqueda. En base a esto, devolvemos los 10 objetos (o menos) más cercanos a la consulta, que estén dentro del radio de búsqueda.

Cabe aclarar que para la búsqueda se utilizó el índice FHQT+ mencionado anteriormente.

El archivo se denomina `10_vecinos_mas_cercanos.sql`

```
CREATE OR REPLACE FUNCTION diez_vecinos_mas_cercanos(img_vector double
precision[], radio NUMERIC) RETURNS SETOF vecino AS
$$
DECLARE
    rec vecino;
    vec record;
    padres bigint[];
    distancia_p numeric;

BEGIN
```



```

padres:=ARRAY(SELECT id_nodo FROM tree WHERE id_padre IS null);

FOR vec IN (SELECT * FROM pivote ORDER BY id_pivote) LOOP
    distancia_p := cosine_distance(img_vector, vec.vector);
--Calculamos la distancia del vector al pivote

    padres:= ARRAY(SELECT id_nodo FROM tree
                        WHERE
numrange(distancia_p-radio,distancia_p + radio, '[') && intervalo_act
                        AND id_padre = ANY(padres));

    END LOOP;

RETURN QUERY (SELECT id, path, id_hoja, web_path,
cosine_distance(vector, img_vector) as distancia FROM imagenes
                WHERE id_hoja = ANY(padres)
                AND cosine_distance(vector, img_vector) <=
radio
                ORDER BY distancia
                LIMIT 10);
END;

$$ LANGUAGE "plpgsql";

CREATE TYPE vecino AS(
id integer,
path VARCHAR(255),
id_hoja bigint,
web_path VARCHAR(2083),
distancia double precision
);

```

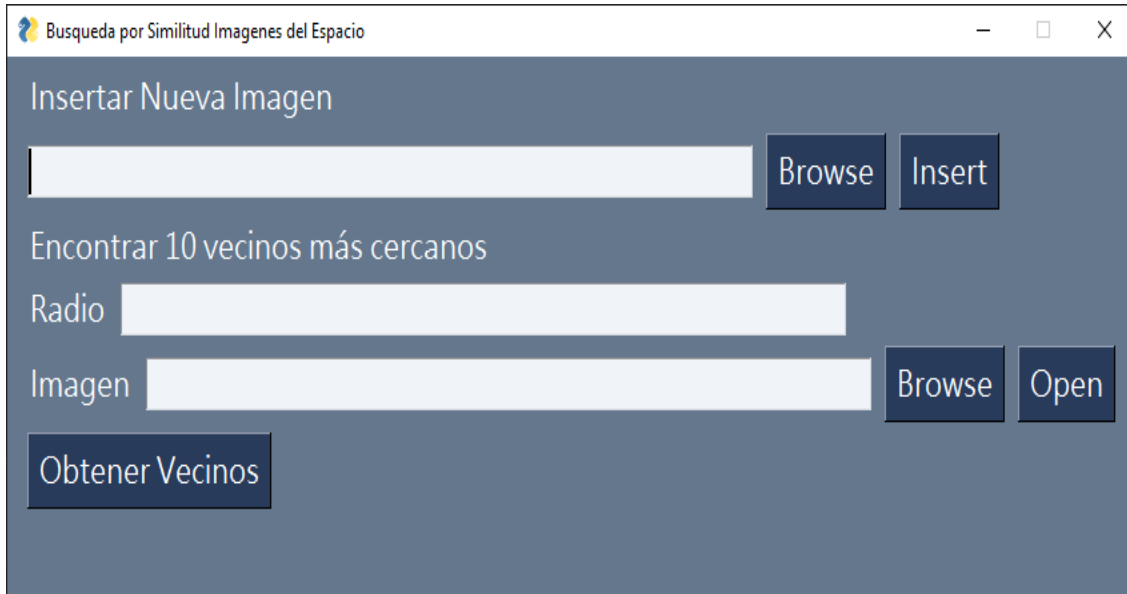
Creación de la aplicación

Para poder insertar imágenes y realizar búsquedas por similitud en la base de datos se creó una aplicación cliente que permite realizar estas dos funciones. Esta aplicación se creó con una librería llamada PySimpleGUI y también se utilizó otra llamada Psycopg2 para poder realizar consultas a la BBDD.

Los archivos utilizados para crear esta aplicación son:

- **app_cliente.py**: contiene la definición de la app y los métodos para insertar imágenes y realizar consultas.
- **config.py**: nos trae los archivos de configuración para acceder a la base de datos.
- **database.ini**: Contiene las credenciales de acceso a la base de datos.

Interfaz de usuario



Busqueda por Similitud Imagenes del Espacio

Insertar Nueva Imagen

Encontrar 10 vecinos más cercanos

Radio

Imagen

A continuación se muestra un ejemplo del segundo método donde primero se le pasa un radio pequeño y nos devuelve una sola consulta, que justamente es la misma imagen que se encuentra en la base de datos, y el segundo ejemplo muestra la misma imagen pero con un radio más grande, lo que hace que el método nos devuelva 10 imágenes.



Encontrar 10 vecinos más cercanos

Radio

Imagen

id	id hoja	distancia
35536	84898	0.0

Encontrar 10 vecinos más cercanos

Radio:

Imagen:

id	id hoja	distancia					
35536	84898	0.0	E:\Proyecto GAD\Imagenes\imagen0.2_0.4.jpg	https://dr12.sdss.org/sas/dr12/boss/photoObj/frames	<input type="button" value="Open"/>		
35584	85108	0.021713398560453623	E:\Proyecto GAD\Imagenes\imagen0.2_17.999999999999998.jpg	https://dr12.sdss.org/sas/dr12/boss/photoObj/frames	<input type="button" value="Open"/>		
38803	94921	0.02538391467925294	E:\Proyecto GAD\Imagenes\imagen3.8000000000000003_22.999999999999995.jpg	https://dr12.sdss.org/sas/dr12/boss/photoObj/frames	<input type="button" value="Open"/>		
37094	90036	0.02603731599648318	E:\Proyecto GAD\Imagenes\imagen1.8_4.4000000000000001.jpg	https://dr12.sdss.org/sas/dr12/boss/photoObj/frames	<input type="button" value="Open"/>		
35556	84987	0.028242352279483796	E:\Proyecto GAD\Imagenes\imagen0.2_12.399999999999998.jpg	https://dr12.sdss.org/sas/dr12/boss/photoObj/frames	<input type="button" value="Open"/>		
38984	95441	0.028352858425926386	E:\Proyecto GAD\Imagenes\imagen4.0_22.799999999999995.jpg	https://dr12.sdss.org/sas/dr12/boss/photoObj/frames	<input type="button" value="Open"/>		
36420	87967	0.02852732654469603	E:\Proyecto GAD\Imagenes\imagen1.2_12.999999999999998.jpg	https://dr12.sdss.org/sas/dr12/boss/photoObj/frames	<input type="button" value="Open"/>		
36430	88000	0.02884615781541855	E:\Proyecto GAD\Imagenes\imagen1.2_14.9999999999999979.jpg	https://dr12.sdss.org/sas/dr12/boss/photoObj/frames	<input type="button" value="Open"/>		
37107	90072	0.02885553569000887	E:\Proyecto GAD\Imagenes\imagen1.8_7.00000000000000036.jpg	https://dr12.sdss.org/sas/dr12/boss/photoObj/frames	<input type="button" value="Open"/>		
36016	86673	0.028984116572043694	E:\Proyecto GAD\Imagenes\imagen0.6000000000000001_6.4000000000000003.jpg	https://dr12.sdss.org/sas/dr12/boss/photoObj/frames	<input type="button" value="Open"/>		

Esta aplicación sin embargo, no es la utilizada para realizar las 50 consultas a la base de datos y obtener los porcentajes de acierto ya que sería difícil e incómodo de hacerlo por una interfaz de usuario. Por lo tanto, decidimos realizar estas consultas a partir de un script que se llama `get_metrics.py` y toma como parámetro un archivo de texto el cual debe contener los nombres de las imágenes a consultar.

Estas imágenes son transformadas agregándoles ruido o rotándolas para después comparar los resultados de una consulta con la imagen original y la modificada. La métricas que estamos calculando son:

- Primer acierto: Indica si ambas consultas devuelven la misma imagen como la imagen más similar.
- Primeros 5 aciertos: Indica cuántas de las primeras 5 imágenes que devuelven ambas consultas son las mismas.
- Precisión: En base a los primeros 5 aciertos, obtenemos el porcentaje de acierto para esa consulta
- Precisión y desviación global: Con la precisión de las consultas, obtenemos el promedio y la desviación estándar global.

Resultados

Se utilizó el radio = 0.3 para todas las consultas

Nro	Imagen	Primer Lugar	Primeros 5 lugares	Porcentaje aciertos primeros 5 lugares
1	imagen3.2000000000000006_0.4.jpg	<input checked="" type="checkbox"/>	Hit 4 of 5	80

2	imagen2.2_1.2.jpg	✗	Hit 4 of 5	80
3	imagen3.8000000000000003_8.8.jpg	✓	Hit 5 of 5	100
4	imagen5.0_28.19999999999932.jpg	✗	Hit 4 of 5	80
5	imagen2.4000000000000004_11.19999999999992.jpg	✓	Hit 4 of 5	80
6	imagen5.6000000000000005_26.99999999999936.jpg	✗	Hit 4 of 5	80
7	imagen2.0_28.9999999999993.jpg	✓	Hit 5 of 5	100
8	imagen3.8000000000000003_15.79999999999976.jpg	✗	Hit 4 of 5	80
9	imagen4.2_36.5999999999998.jpg	✓	Hit 4 of 5	80
10	imagen4.6000000000000005_30.19999999999925.jpg	✗	Hit 4 of 5	80
11	imagen5.4_36.19999999999974.jpg	✓	Hit 4 of 5	80
12	imagen2.4000000000000004_25.19999999999942.jpg	✓	Hit 4 of 5	80
13	imagen2.0_31.3999999999992.jpg	✓	Hit 4 of 5	80
14	imagen4.4_19.9999999999996.jpg	✗	Hit 4 of 5	80
15	imagen0.2_17.99999999999968.jpg	✓	Hit 4 of 5	80
16	imagen2.6000000000000005_32.1999999999992.jpg	✓	Hit 4 of 5	80
17	imagen4.4_29.99999999999925.jpg	✗	Hit 4 of 5	80
18	imagen3.2000000000000006_12.79999999999986.jpg	✗	Hit 4 of 5	80
19	imagen0.2_12.39999999999988.jpg	✓	Hit 5 of 5	100
20	imagen4.2_22.5999999999995.jpg	✓	Hit 4 of 5	80
21	imagen3.2000000000000006_22.5999999999995.jpg	✓	Hit 4 of 5	80
22	imagen2.0_34.3999999999995.jpg	✗	Hit 4 of 5	80
23	imagen3.0000000000000004_23.3999999999995.jpg	✓	Hit 4 of 5	80
24	imagen4.6000000000000005_12.59999999999987.jpg	✗	Hit 4 of 5	80
25	imagen2.0_33.39999999999935.jpg	✓	Hit 4 of 5	80
26	imagen4.4_31.1999999999992.jpg	✗	Hit 4 of 5	80
27	imagen3.8000000000000003_27.19999999999935.jpg	✓	Hit 4 of 5	80

28	imagen3.2000000000000006_14.39999999999998.jpg	✓	Hit 4 of 5	80
29	imagen2.2_28.199999999999932.jpg	✓	Hit 4 of 5	80
30	imagen4.4_0.8.jpg	✗	Hit 4 of 5	80
31	imagen3.4000000000000001_8.2000000000000003.jpg	✓	Hit 4 of 5	80
32	imagen3.2000000000000006_15.599999999999977.jpg	✗	Hit 4 of 5	80
33	imagen4.8000000000000001_19.99999999999996.jpg	✓	Hit 5 of 5	100
34	imagen3.8000000000000003_16.199999999999974.jpg	✗	Hit 4 of 5	80
35	imagen2.2_18.199999999999967.jpg	✓	Hit 5 of 5	100
36	imagen2.2_29.19999999999993.jpg	✓	Hit 4 of 5	80
37	imagen4.4_1.799999999999998.jpg	✓	Hit 4 of 5	80
38	imagen2.2_5.2000000000000002.jpg	✓	Hit 4 of 5	80
39	imagen2.0_29.19999999999993.jpg	✓	Hit 5 of 5	100
40	imagen4.0_22.79999999999995.jpg	✓	Hit 4 of 5	80
41	imagen3.0000000000000004_18.799999999999965.jpg	✓	Hit 4 of 5	80
42	imagen2.2_1.4.jpg	✗	Hit 4 of 5	80
43	imagen2.2_8.0000000000000004.jpg	✓	Hit 4 of 5	80
44	imagen5.2_30.999999999999922.jpg	✗	Hit 4 of 5	80
45	imagen2.2_0.8.jpg	✓	Hit 4 of 5	80
46	imagen3.4000000000000001_15.399999999999977.jpg	✓	Hit 4 of 5	80
47	imagen3.4000000000000001_15.199999999999978.jpg	✓	Hit 4 of 5	80
48	imagen3.4000000000000001_16.99999999999997.jpg	✓	Hit 4 of 5	80
49	imagen2.6000000000000005_28.59999999999993.jpg	✓	Hit 4 of 5	80
50	imagen3.6_19.99999999999996.jpg	✗	Hit 4 of 5	80
-	Porcentaje Primer Acierto	66 %	Porcentaje de aciertos en los primeros 5 Total	82.4 %