**Proyecto 3:**

**Base de Datos Multimedia**

1. **Introducción**

El logro del estudiante está enfocado a entender y aplicar los algoritmos de búsqueda y recuperación de la información basado en el contenido.

Este proyecto está enfocado al uso una estructura multidimensional para dar soporte a las búsqueda y recuperación eficiente de imágenes en un servicio web de reconocimiento facial.

1. **Backend**: Servicio Web de Reconocimiento Facial

Implementar un web service para la identificación automática de personas a partir de una colección grande de imágenes de rostros.

El procedimiento general consiste en lo siguiente:

* Extracción de características
* Indexación de vectores característicos para búsquedas eficientes
* Algoritmo de búsqueda KNN

**2.1. Extracción de características**

Para la extracción de características se usará la librería [Face\_Recognition](https://github.com/ageitgey/face_recognition). En dicha librería ya se encuentra implementado las técnicas necesarias para obtener de cada imagen una representación compacta y representativa del rostro (enconding). El tamaño del vector característico es de 128 ([ver referencia](https://www.pyimagesearch.com/2018/06/18/face-recognition-with-opencv-python-and-deep-learning/)). La efectividad del reconocimiento ha sido probada con modelos de búsqueda basados en *deep learning* (99.38% de precisión). En este proyecto vamos a usar solo las **funciones básicas** de dicha librería: **face\_encodings y face\_distance**.

Se usará una colección de referencia con más de 13 mil imágenes de rostros de personas, disponible en el [siguiente enlace](http://vis-www.cs.umass.edu/lfw/). Algunas personas tienen más de una imagen asociada, considere todas.

El grupo puede optar por otro dataset de rostros [disponibles en la web](https://www.kaggle.com/c/deepfake-detection-challenge/discussion/121594).

**2.2. Indexación y Búsqueda**

* ***KNN-Secuencial:*** Implementación de los algoritmos de búsqueda sin indexación
  + Búsqueda KNN con cola de prioridad, el cual recibe como parámetro el objeto de consulta y la cantidad de objetos a recuperar K.
  + Búsqueda por Rango, el cual recibe como parámetro el objeto de consulta y un radio de búsqueda. Incluir el análisis de la distribución de la distancia para experimentar con 3 valores de radio diferente.
* ***KNN-RTree:*** Para búsquedas eficientes, hacer uso de una librería de índice espacial para indexar todos los vectores característicos que serán extraídos de cada imagen de la colección. Opciones:
  1. [R-Tree - C++](https://github.com/nushoin/RTree)
  2. [R-Tree - Python](https://rtree.readthedocs.io/en/latest/tutorial.html)
  3. [GiST index - PostgreSQL](https://medium.com/postgres-professional/indexes-in-postgresql-5-gist-86e19781b5db)
* ***KNN-HighD:*** Debe considerar el hecho de que un índice para espacios vectoriales reduce su eficiencia con dimensiones muy altas ([maldición de la dimensionalidad](https://bib.dbvis.de/uploadedFiles/190.pdf)). Estudie [como mitigar este problema](https://www.baeldung.com/cs/k-nearest-neighbors) y aplique alguna de estas soluciones:

1. [**PCA**](https://towardsdatascience.com/pca-using-python-scikit-learn-e653f8989e60)
2. [**KD-Tree**](https://en.wikipedia.org/wiki/K-d_tree)
3. [Locality Sensitive Hashing](https://graphics.stanford.edu/courses/cs468-06-fall/Slides/aneesh-michael.pdf) (LSH)
4. [**Faiss (GPU & HNSW)**](https://engineering.fb.com/2017/03/29/data-infrastructure/faiss-a-library-for-efficient-similarity-search/)índice basado en grafos y soporte para GPU

**2.3. Experimento:**

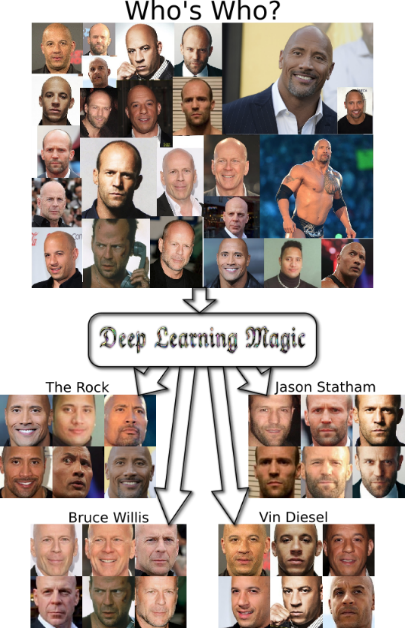
Ejecute el KNN-RTree, KNN-secuencial y el KNN-HighD sobre una colección de objetos de tamaño N y compare la eficiencia en función del tiempo de ejecución.

|  |  |  |  |
| --- | --- | --- | --- |
|  | KNN-Secuencial | KNN-RTree | KNN- HighD |
| N = 100 |  |  |  |
| N = 200 |  |  |  |
| N = 400 |  |  |  |
| N = 800 |  |  |  |
| N = 1600 |  |  |  |
| N = 3200 |  |  |  |
| N = 6400 |  |  |  |
| N = 12800 |  |  |  |

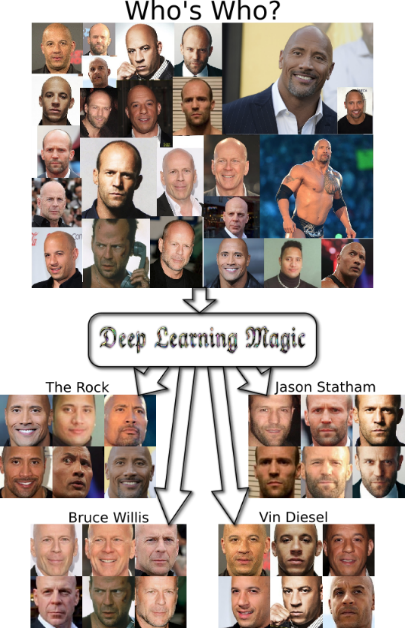
\* Mantener el valor de K = 8.

1. **Frontend:** Motor de Búsqueda

Construir una aplicación web adaptativa que permita interactuar con el Web Service de reconocimiento facial. La consulta es cualquier imagen que incluya un rostro (puede usar fotos externas) y debe retornar el top-k de los personajes más parecidos.



WS Face Recognition



* Muestre los resultados de búsqueda interactivamente.
  + El rostro de consulta y el valor de k debe ser un dato de entrada.

1. **Entregable**

* Los alumnos formaran grupos de máximo de tres integrantes.
* El proyecto estará alojado enteramente en GitHub, GitLab o Bitbucket.
* Trabajar de forma colaborativa, se considerará para su nota individual.
* Incluir en el informe un cuadro de actividades por integrante en [Project Boards](https://github.com/features/issues).
* En el Canvas subir solo el **enlace público** del proyecto.
* La fecha límite de entrega es la semana 16 (no habrá prorroga).

1. **Informe del proyecto**

* Archivo Readme o Wiki
* Ortografía y consistencia en los párrafos.
* El informe debe describir todos los aspectos importantes de la implementación.
  + Librerías utilizadas
    - Describa la técnica de indexación de las librerías utilizadas
    - Como se realiza el KNN Search y el Range Search (si es que lo soporta)
  + Análisis de la maldición de la dimensionalidad y como mitigarlo
    - Incluir imágenes/diagramas para una mejor comprensión.
  + Experimentación
    - Tablas y gráficos de los resultados
    - Análisis y discusión
* En la semana 16 cada grupo **presentará su producto**, en donde se visualice el programa en acción. La duración de la exposición no debe exceder 10 minutos. ¡¡Venda su producto!!
  + Los resultados deben visualizarse de forma amigable e intuitiva para el usuario.

1. **Rúbrica**

Va adjunto