Ejercicio 1
Script 1
El directorio script_1, contiene los archivos necesarios para indexar la colección, graficar la distribución de tamaño de las postings, calcular overhead de la colección y por cada documento.

El programa se corre ejecutando el archivo run.py, sin parámetros. "python3 run.py"
Los parámetros al programa se modifican en el archivo "constants.py", donde se encuentran los siguientes:

- EMPTY_WORDS_PATH, path al archivo de palabras vacías.
  Utilizar el valor None para no extraer palabras vacías

- DIRPATH, path al directorio donde se encuentra el corpus.

- MIN_TERM_LENGTH y MAX_TERM_LENGTH tamaño maximo y minimo para que un token pueda ser considerado un término

- STRING_STORE_CRITERION, utilizando el valor "MAX" la cantidad de caracteres que serán utilizados para grabar los términos y los nombres de los documentos en sus archivos correspondientes, serán equivalentes al tamaño del término o documento más largo.
  Es decir, ningún término ni nombre de documento será recortado, con la desventaja de muchos bits desperdiciados, y un incremento en el overhead.
  Utilizando el valor "STATIC" se utilizarán los valores definidos en DOCNAMES_SIZE y TERMS_SIZE

- STEMMING_LANGUAGE, lenguaje utilizado por el Stemmer. Utilizar None para no realizar stemming.

- EXTRACT_ENTITIES, acepta valores "True" o "False", indica si se deben evaluar o no expresiones regulares para extraer entidades.

- HTML_FILES, acepta valores "True" o "False", indica si se deben extraer texto legible de archivos html, para evitar indexar tags.

- CORPUS_FILES_ENCODING, encoding del corpus. Utilizo "ISO-8859-1" para la colección Wiki-Small que pasé a txt, y el encoding es el dicho anteriormente, sino, el resto es "UTF-8".

- ID_IN_DOCNAME, acepta valores "True" o "False", indica si se debe extraer el id del nombre del documento. Utilizado en colecciones de prueba donde el nombre de los documentos incluyen el id como "doc120.txt"

- COMPUTE_OVERHEAD, acepta valores "True" o "False", indica si se debe calcular el overhead.

- PLOT_RESULTS, acepta valores "True" o "False", indica si se deben plotear los resultados.

- WORKERS_NUMBER, cantidad de threads que se utilizan para realizar la indexación.

- Los demás se tratan de constantes donde se define el path donde se exportan los archivos binarios, path donde se exportan los archivos txt y png y nombres de los archivos.

En la carpeta output se exportan los archivos resultantes, en human_files, los archivos en formato txt para debug, y los png de los plot realizados.
En la carpeta index_files, se guardan los archivos binarios. (El vocabulario, el índice invertido, el archivo de nombres de documentos mapeados a ids, y el archivo de metadata). Dicho archivo de metadata indica el tamaño utilizado para almacenar los nombres de los documentos, como también para los términos, el lenguaje de stemming utilizado, y si se extrajeron entidades.

Script 2
El directorio script_2 posee el script que permite recuperar la posting de un término dado. Se ejecuta con el archivo run.py. Se definen los parámetros en el archivo constants.py.

- INDEX_FILES_PATH, directorio donde se encuentran los archivos binarios.
- METADATA_FILE, nombre del archivo de metadatos.
- BIN_VOCABULARY_FILEPATH, BIN_INVERTED_INDEX_FILEPATH, BIN_DOCNAMES_IDS_FILEPATH, nombre del archivo del vocabulario, índice invertido y docnames_ids.

En caso de no haber modificado el archivo de constantes del script anterior, no es necesario modificar ninguno de estos.

Test_results.py
Si con el script 1 se indexó tanto la colección "collection_test" como "collection_test_ER2" se puede utilizar el script test_results.py, indicando en la constante que tiene definida RESULTS_FILE, el json que posee la información de la colección, y utilizando el script_2 para recuperar las postings de todos los términos, verifica que estas coincidan con las del archivo json.
Si se utiliza collection_test, EXTRACT_ENTITIES debe setearse el False.
Mientras que si se utiliza collection_test_ER2 EXTRACT_ENTITIES debe establecerse en True.

Ejercicio 2
Similar al ejercicio anterior.
Se ejecuta con "python3 run.py"
El archivo de constantes es "constants.py", utiliza el índice del ejercicio 1, además, se pueden definir en el archivo de constantes los símbolos para AND, OR y NOT.

Test_results.py

Si en el ejercicio 1, en el script 1, se indexaron las colecciones "collection_test" o "collection_test_ER2", definiendo el archivo json en RESULTS_FILE, se realizan todas las posibles queries aceptados por el programa que son:
Queries |q| = 2

- t1 AND t2
- t1 OR t2
- t1 NOT t2

Queries |q| = 3

- t1 AND t2 AND t3
- (t1 OR t2) NOT t3
- t1 AND t2) OR t3

Por un lado, se resuelven con el script de este ejercicio (ejercicio_2/run.py), y por el otro, con operaciones de conjuntos sobre los doc_ids del archivo json.

Ejercicio 3
Utiliza un archivo de constantes similar a los anteriores, solo que agrega el path al archivo donde se encuentran las querys, QUERYS_FILE_PATH.

Generate_statistics.py
Realiza todas las consultas parseando el archivo de queries, utilizando retrievals que usan el índice invertido de disco, como también, cargado en memoria. Luego exporta dichos tiempos en un archivo json llamado statistics.json, que es procesado por el siguiente script. Además, verifica que las respuestas de ambos retrieval sean las mismas.

Process_statistics.py
Procesa las estadísticas del archivo json, calcula y muestra promedios, realiza plots.

Ejercicio 4
Script 1

El directorio script_1, contiene los archivos necesarios para indexar la colección, similar al ejercicio 1, agrega al archivo de constantes el parámetro DOCUMENT_LIMIT que es el valor que indica cada cuántos documentos se debe hacer el volcado a disco.

Se corre de la misma forma que el ejercicio 1, "python3 run.py"

Script 2
Script que permite mostrar las postings de un término dado, tiene el mismo archivo de constantes que el script 2, del ejercicio 1, se corre de la misma forma,

Test_results.py
Verifico como en los ejercicios anteriores, cuando se usan las colecciones collection_test, si las postings de todos los términos del vocabulario, coinciden con las del archivo json.

Ejercicio 5
Script 1

Mismo archivo de constantes que los ejercicios anteriores, ejecución con el archivo run.py. Se agrega una constante que indica el nombre donde se va a almacenar la norma de los documentos, BIN_NORM_FILEPATH.

Script 2
Utiliza un archivo de constantes muy similar a los "script_2" de los ejercicios anteriores.

Test_results.py
También se verifican resultados de forma similar a los ejercicios anteriores.

Ejercicio 6
Script 1
Archivo de constantes constants.py, ejecución con python3 run.py. Se agrega el parámetro para indicar la ubicación del archivo de posiciones BIN_POSITIONS_FILEPATH.

Script 2
Igual que los ejercicios anteriores, se agregan constantes que indican la sintaxis de los operadores.

Ejercicio 7
Script 1
Misma ejecución y definición de constantes que los ejercicios anteriores. Se agrega una constante que indica el K de las skips.

Script 2
Muy similar a los "script_2" de ejercicios anteriores.

Ejercicio 8
Script 1
Ejecución mediante el archivo "parse_dump.py". Se utiliza el archivo de constantes para definir el path a la colección dump10k, el k de skips, y los nombres de los archivos.

Script 2
Ejecución similar a todos los ejercicios anteriores, "python3 run.py"

Los **ejercicios 9, 10, y 11**, no tienen interfaz para consultar postings o algo similar, de forma interactiva.