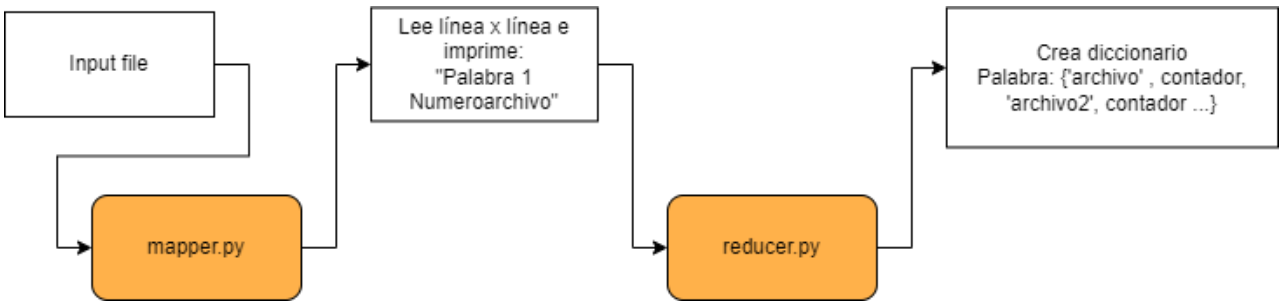
Tarea 3 Sistemas Distribuidos
Javier Guerrero - Darío Marmie

1. **Wikipedia-Api**
   El código de la conexión con la api de Wikipedia se encuentra en el siguiente Link.
   En este, primeramente se definen los parámetros a trabajar con la conexión, esto puesto que este código obtiene 10 artículos de manera aleatoria. (fuente: Random)
   Luego, se obtienen los títulos de estos artículos y se guardan indexados en un documento de texto.
   De igual manera, se extrae el texto dentro de estas páginas mediante la API y se guardan en archivos de textos por separados siendo el nombre de estos números del 1 al 10
2. **Carpeta 1 y 2 (opcional)**
   Dentro de las carpetas 1 y 2 se encuentran los archivos enumerados desde el 1-5 en la carpeta 1, y del 6-10 en la carpeta 2. Esto se solicitaba en el enunciado de la tarea, más no tenía puntaje por lo que se entendió que era opcional. De todas formas se realizó, y estas carpetas luego se envían al sistema de archivos de Hadoop para poder utilizarlas como input.
3. **MapReduce y Wordcount**
   Para contar palabras dentro de un código de MapReduce se debe utilizar un trabajo de WordCount. Lo que principalmente realiza un trabajo de Word Count es tomar los datos, separarlos en bloques, y reordenar los outputs e inputs para reducir el número de tareas. En Hadoop, existe el Hadoop Distributed File System, que permite a todos los nodos pertenecientes al clúster acceder al mismo directorio de archivos. En dicho directorio se guardan los inputs y outputs de un trabajo.

   En el caso de esta tarea, se utilizó un proceso de MapReduce que realizaba el conteo en base al archivo que se le entregaba. El Mapper entregaba una palabra, un 1 y el número del archivo del que se está leyendo (Dicho número de archivo se encontraba en la primera línea de los .txt a procesar). Luego, el Reducer toma todas las palabras y genera un diccionario donde asocia el archivo y el contador a la palabra respectiva.

   La idea detrás del uso de MapReduce y de WordCount especialmente en Hadoop, es para poder dividir la carga que se genera cuando se trabaja con un gran volúmen de datos, pudiendo repartir los trabajos a través de los distintos nodos que pertenezcan al clúster de Hadoop. Esto se logra a través del uso de YARN (Yet Another Resource Negotiator), que permite la distribución de recursos en conjunto con el Resource Manager.



4. **Base de datos (opcional)**
   Para este caso, también se solicitaba opcionalmente el uso de una base de datos para poder desarrollar el buscador. No se encontró que fuera necesario el uso de una base de datos, por lo que se decidió utilizar sólo archivos de textos y trabajar en base a diccionarios de Python para lograr el desarrollo de las búsquedas.
5. **Buscador**
   El buscador se puede encontrar en el siguiente código, en este primeramente se obtienen los índices y los títulos de los artículos para su posterior uso.
   Luego, se realiza la lectura del archivo obtenido por el procesamiento de hadoop, el cual se encargará de contar cuantas veces se encuentran las palabras buscadas por cada artículo, esto para todos las palabras ingresadas. Posteriormente, seleccionará al artículo que posee más coincidencias, se le obtendrá el nombre de este mediante su index, y con ello, se realizará la conexión a la API de wikipedia con el fin de obtener el link.
6. **Video (click en la imagen)**