ST0263 - Proyecto 4
Clustering de documentos a partir de métricas de similitud basado en Big Data

Juan Carlos Estrada Alvarez
Universidad EAFIT
jestra52@eafit.edu.co

Valentin Quintero Castrillon
Universidad EAFIT
vquinte3@eafit.edu.co

Abstract

During the history of computing there have always been problems of computing that are limited to the times required to solve them due to the great power of compute that they demand. From this context arises data-intensive computing problems.

Knowing that currently most of the computers and all the centers of super compute of the world have multiple processors and coprocessors, the parallelization of problems is a very striking option to shorten execution times, and specifically, The Big Data environment is perfect for implementing solutions to data-intensive problems.

In this document we will present an implementation to a problem of document Clustering in a Big Data environment, and we will show differences of a solution made for parallelization with MPI vs a solution made with Spark for Hadoop.

Keywords Spark, K-means, Paralelo, Tiempos, Agrupamiento, Clustering, Big Data, Similaridad, Python, Hadoop, MPI.

1. Introducción

La infinita curiosidad del ser humano lo ha llevado a lograr grandes avances científicos y tecnológicos en el transcurso de toda su historia. En tiempos modernos, el ser humano se ha visto obligado a hacer uso de la computación para poder extraer y analizar grandes cantidades de información, donde se requiere computar dado que no es posible para el ser humano almacenar y analizar tanta información en tan poco tiempo. De ahí sale la ciencia de la información; la informática. Se inventaron las computadoras principalmente para crear una solución viable; el procesamiento, el almacenamiento y el análisis de la información.

El problema ahora es que cada vez requerimos más capacidad de computo para poder almacenar y ver resultados en un tiempo relativamente corto, y las soluciones típicas, usualmente pensadas en forma serial, se quedan obsoletas. Es por eso que en los ultimos años, la extracción y almacenamiento de información con las soluciones seriales típicas se han vuelto casi imposibles, debido a los gigantescos requisitos de hardware y procesamiento que conlleva hacer este proceso. Para este tipo de problemas se hace uso de los ambientes Big Data, que nos permite computar los problemas en tiempos relativamente cortos y, al mismo tiempo, almacenar y extraer datos de grandes clusteres de computadoras dedicados a esto.

En este documento nos enfocaremos en uno de esos problemas; El análisis de similitud entre documentos, donde se separan dichos docuentos en clusters (grupos), específicamente mediante un sistema de separación de archivos por similitud.

Comenzaremos implentando una solución en Spark en un ambiente Hadoop y su ejecuión en terminos de eficiencia, para luego compararla con una solución paralela pensada para MPI.

2. Marco teórico

La minería de texto (text mining), es una de las técnicas de análisis de textos que ha permitido implementar una serie de aplicaciones muy novedosas hoy en día. Buscadores en la web (Google, Facebook, Amazon, Spotify, Netflix, entre otros), sistemas de recomendación, procesamiento natural del lenguaje, son algunas de las aplicaciones.

Las técnicas de agrupamiento de documentos (clustering) permiten relacionar un documento con otros parecidos de acuerdo a alguna métrica de similaridad. Esto es muy usado en diferentes aplicaciones como: Clasificación de nuevos documentos entrantes al dataset, búsqueda y recuperación de documentos, ya que cuando se encuentra un documento seleccionado de acuerdo al criterio de búsqueda, el contar con un grupo de documentos relacionados, permite ofrecerle al usuario otros documentos que potencialmente son de interés para él.

Algoritmos de agrupamiento particional han sido reconocidos como más adecuados en comparación con los esquemas de agrupación jerárquico para procesar grandes conjuntos de datos, utilizando este método como respuesta a una problemática con la que nos enfrentamos claramente en la actualidad. Un volumen de datos muy grande que crean desafios de organización efectiva de textos.

En la creación de estos algoritmos se parte siempre con un vector de palabra. Este es sencillamente cada palabra del documento almacenada en un arreglo, además se debe tener cuenta que en el idioma inglés existen al menos 550 palabras consideradas "STOPWORDS" las cuales no dan relevancia al texto, por ejemplo "a, the, do". Estas pueden estar 1000 veces entre todos los textos pero no aportan ninguna característica diferenciadora.

La manera en la que se busca una similitud entre documentos en esta práctica está dada por el peso de una palabra el cual, se asigna bajo unos criterios específicos como la distancia eucladiana y la similitud del coseno.

Por otro lado, no todas las medidas para la distancia pueden ser consideradas métricas. Estas tienen que cumplir 4 requisitos. Sean x, y los dos objetos en un conjunto y $d(x, y)$ sea la distancia entre x, y.

- La distancia entre dos puntos cualesquiera debe ser no negativa, es decir, $d(x, y) = 0$.
- La distancia entre dos objetos debe ser cero si y solo si los dos objetos son idénticos, es decir $d(x, y) = 0$ si y solo si $x = y$.
- La distancia debe ser simétrica, es decir, la distancia desde x a y es lo mismo que la distancia de y a x, así que $d(x, y) = d(y, x)$.
- La medida debe satisfacer la desigualdad del triángulo que es $d(x, z) < d(x, y) + d(y, z)$.

2.1. Distancia Euclidiana

Mide la distancia ordinaria entre dos puntos y puede se puede medir fácilmente con una regla en dos o tres dimensiones espacio, que es ampliamente utilizada en problemas de agrupación además de que es el predeterminado para la medida de distancia utilizada con el algoritmo K-means.

$$D_E(\vec{t_a}, \vec{t_b}) = (\sum_{t=1}^{m} |w_{t,a} - w_{t,b}|^2)^{\frac{1}{2}} \qquad (1)$$

donde el término establecido es $T = t_1, ..., t_m$.

Como se menciona anteriormente, utilizamos el valor de $tfidf$ como ponderaciones de terminos, es decir wt, $a = tfidf(da, t)$.

2.2. Similitud del coseno

Representando como vectores de términos los documentos, la similitud corresponde a la correlación entre los vectores. Esto se cuantifica como el coseno del ángulo entre vectores.

Dados dos documentos t_a y t_b, su similitud coseno es:

$$SIM_C(\vec{t_a}, \vec{t_b}) = \frac{\vec{t_a}.\vec{t_b}}{|\vec{t_a}| \times |\vec{t_b}|} \qquad (2)$$

donde t_a y t_b son vectores m dimensiones en el conjunto de términos $T = t_1, ..., t_m$.

Cada dimensión representa un termino con su peso en el documento, que no es negativo. Como resultado, la similitud del coseno es no negativa y limitada entre $[0, 1]$.

Una propiedad importante de la similitud del coseno es su independencia de la longitud del documento.

2.3. Coeficiente de Jaccard

El coeficiente de Jaccard mide la similitud como la intersección dividido por la unión de los objetos.

Para el documento de texto, el coeficiente de Jaccard compara la suma de peso de los términos compartidos al peso de la suma de los términos que están presentes en cualquiera de los dos documentos, pero no son los términos compartidos.

Dados dos documentos t_a y t_b:

$$SIM_J(\vec{t_a}, \vec{t_b}) = \frac{\vec{t_a}.\vec{t_b}}{\left|\vec{t_a}\right|^2 + \left|\vec{t_b}\right|^2 - \vec{t_a}.\vec{t_b}} \qquad (3)$$

El coeficiente de Jaccard es una medida de similitud y rangos entre 0 y 1. Es 1 cuando el $t_a = t_b$ y 0 cuando $t_a y t_b$ son disjuntos; en otras palabras, 1 significa que los dos objetos son lo mismo y 0 significa que son completamente diferentes.

2.4. TF-IDF

TF-IDF son las siglas en inglés de "Term frequency – Inverse document frequency", lo que al traduce a el español "Frecuencia de términos – Frecuencia inversa del documento", y el peso de TF-IDF es un peso que a menudo se utiliza en la recuperación de información y minería de texto. Esta es una medida estadística utilizada para evaluar la importancia de una palabra para un documento en una colección.

La importancia depende completamente del número de veces que aparezca una palabra en un documento, y esto se compensa con la frecuencia de la palabra en el mismo. K-means y sus variaciones son ampliamente utilizadas en el campo ya que estas son elementos centrales en buscadores y todo tipo de aplicaciones en minería de datos.

La suma de TF-IDF es una de las funciones de clasificación más sencillas. Se calcula sumando el TF-IDF para cada termino de consulta; además de que existen un montón de funciones de clasificación que son variantes de este.

Este método puede ser utilizado con éxito para el filtrado de palabras vacías en los diferentes campos temáticos, como el resumen o la clasificación de texto.

TF-IDF se emplea en el campo de información y minería de texto ya que estos usan elementos tales como las bibliotecas digitales y que esta directamente relacionados con los buscadores, que utilizan variaciones de este algorítmo en sus procesos como induración, posicionamiento y la muestra de contenido específico a los usuarios.

Los componentes de este algorítmo son:

- TF: este es la frecuencia en la que un término aparece en un documento, así que en pocas palabras este es el numero de veces en el que una palabra aparece en un texto. Entre más alto sea el TF para un término, más relevante será este para ese documento.

$$TF(i) = \frac{log_2(Freg(i,j)) + 1}{log_2(L)} \qquad (4)$$

- IDF: Es el encargado de disminuir el peso de los términos que se repiten mucho en los documentos, dándole asi mayor valor a las que tienen una menor frecuencia. En otras palabras, es la frecuencia de documentos que contienen un término específico.

$$IDF_t = log(1 + \frac{N_D}{f_t}) \qquad (5)$$

Las formulas anteriores son las utilizadas para calcular la relevancia de un documento al compararse con los demás documentos que compartes palabras claves. Para que los resultados obtenidos sean realmente útiles se debe calcular para todas las palabras relevantes en un texto, además que entre más grande sea la base de datos usada para el cálculo más precisos son los resultados obtenidos.

Una desventaja importante a la hora de usar este algorítmo es que hay que considerar que este no contempla la posibilidad de que los términos evaluados aparezcan agrupados, que se apliquen

normas de lexema o que se estén usando sinónimos.

2.5. K-means

La agrupación en K-means se utiliza cuando se tienen datos sin categoría o grupos definidos. Su objetivo es encontrar conjuntos de datos, que cumplan con el numero de grupos definidos por k.

Este es un algoritmo iterativo para asignar cada punto de datos a los correspondientes grupos de k, cumpliendo con las características proporcionadas.

Los datos son agrupados según la similitud de las características. Los resultados que arroja el algoritmo de K-means son:

- Los centroides de los clusters k, los cuales se pueden utilizar para hallar nuevos clusters.
- Etiquetas para los datos de entrenamiento (cada punto de datos se asigna a un solo grupo).

La función de K-means está dada por la ecuación:

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| {x_i}^j - c_j \right\|^2 \tag{6}$$

donde $\left\| {x_i}^j - c_j \right\|^2$ es una medida de distancia escogida entre un punto de datos y el centro del cluster; este indica la distancia que se encuentra en los puntos de datos y sus respectivos clustres.

El algoritmo se compone de los siguientes pasos:

- Paso de asignación de datos: En este paso cada centroide define uno de los clusters, Y luego cada punto de datos se asigna a su centroide más cercano, llendo en función de la distancia asignada.
- Paso de actualización del centroide: Esto se hace tomando la media de todos los puntos de datos asignados al clúster de ese centroide.

Se repiten los pasos hasta que los centroides ya no se muevan. Produciendo una separación de los objetos en grupos a partir de los cuales se puede calcular la métrica que se va a minimizar.

Se debe tener en cuenta que:

- Se garantiza que este algoritmo converge a un resultado.
- Aunque se puede demostrar que el procedimiento siempre terminará, el algoritmo de K-means no necesariamente encuentra la configuración más óptima, que corresponde al mínimo de la función objetivo global.
- El algoritmo también es significativamente sensible a los centros de cluster iniciales seleccionados aleatoriamente.
- El algoritmo K-means se puede ejecutar varias veces para reducir el anterior efecto.

3. Análisis y diseño mediante ETL

3.1. Problema

Leer una gran cantidad de ficheros.

Cada documento (fichero) es representado por un RDD (tipo de dato en Spark), en el cual se eliminan las denominadas “STOPWORDS” del idioma inglés, las cuales son palabras que se repiten mucho y son irrelevantes a la hora de determinar la similaridad entre documentos.

Utilizamos el algorítmo ETL (Extract, Transform, Load). Esto permite facilitar el movimiento de los datos y la transformación de los mismos, para poder analizarlos y utilizarlos con un objetivo.

Se tienen entonces los siguientes pasos:

- Extracción: Se extrae la información de un grupo de documentos en un dataset, sea localmente o desde el cluster Hadoop (HDFS).
- Transformación: La transformación de la información cuando se separan los términos de un documento y se quitan las STOPWORDS de dicho documento.
- Carga: La carga se da cuando se tiene el resultado y se prepara para almacenar o cargar en algún directorio, localmente o en el cluster Hadoop (HDFS). También es posible almacenarlos en alguna base de datos, pero en la solución implementada no damos esta opción.

3.2. Partición

- Lectura de ficheros, sea local o desde el cluster Hadoop (HDFS).
- Simplificacion del contenido de los archivos con base en STOPWORDS.

- Calcular la distancia entre documentos.
- Agrupacion de documentos mediante la aplicacion de K-Means.

3.3. Comunicación

Para la comunicación se tienen las tecnologías del ecosistema distribuido Hadoop:

- Spark.
- HDFS.
- YARN.
- MapReduce.

3.4. Aglomeración

- Lectura de ficheros, simplificacion y eliminación.
- Selección de palabras más repetidas, mapearlas y computarlas mediante TF-IDF.
- Agrupacion con K-Means.

4. Implementación en Pyspark

Para llevar a cabo la implementación del proyecto en el framework Spark, se utilizaron las estructuras de datos y funcionalidades proporcionadas por este framework(RDD, DataFrame), a continuacion se explicará a detalle como y en que momento fueron utilizados estos aspectos para el desarrollo.

Lectura del dataset: Para llevar a cabo la lectura del dataset, se utilizó la función *wholeTextFiles* la cual nos es proporcionada por el SparkContext, esta función recibe la ruta que corresponde al directorio del dataset dentro del cluster Hadoop. Esta fúncion realiza la lectura de los ficheros dentro del directorio proporcionado y retorna una estructura de datos llamada *RDD* proporcionada por Spark.

RDD (Resilient Distributed Dataset): Es el modo de abstracción primario en Apache Spark y el nucleo de Spark. Un RDD es una colección distribuida y resiliente de registros propagada en una o muchas particiones(nodos).

```
# Lee todos los archivos dentro del directorio especificado y obtiene un RDD
text_files = sc.wholeTextFiles(path)
```

Representación de RDD a DataFrame: Luego de obtener el RDD, este pasa a ser convertido en un DataFrame, que es otro tipo de dato brindado por Apache Spark, este tipo de dato se puede asemejar con las tablas en SQL ya que la forma en como funcionan es a traves de columnas y tuplas. Otro aspecto que hace que se parezca a una tabla en SQL, es que se pueden realizar consultas sobre un DataFrame.

En la porción de código que aparece acontinuacion, se puede observar que se obtiene un DataFrame por medio del RDD creado previamente, en el cual se obtienen dos campos que corresponden con la ruta del documento, y el contenido de este.

```
# Se convierte el RDD obtenido a un DataFrame
df = text_files.toDF(["Ruta", "Documento(String)"])
```

Representación de documentos: Cada documento del dataset es representado por un arreglo de palabras que fueron obtenidas a partir del contenido de este documento. Para obtener este arreglo se usa la clase *Tokenizer*
de Spark, la cual proporciona la funcionalidad para transformar una cadena de texto en una lista de terminos que corresponden a las palabras de este texto. La clase Tokenizer logra esto aplicando la función de transformación sobre los datos de una columna específica de un DataFrame, y generando un DataFrame con una nueva columna que corresponde a los tokens de cada tupla de DataFrame.

```
# Crea los tokens que corresponden a las palabras de los documentos
tokenizer = Tokenizer(inputCol="Documento(String)", outputCol="Tokens")
tokenized = tokenizer.transform(df)
```

Eliminación de StopWords: Apache Spark proporciona una clase llamada StopWordsRemover, que contiene la funcionalidad para eliminar las palabras repetitivas e irrelevantes de un conjunto de terminos. Esta clase proporciona la funcionalidad de transformación que se aplica sobre una columna específica de un DataFrame, generando asi un nuevo DataFrame con una columna adicional que contiene el conjunto de terminos con las StopWords eliminadas.

```
remover = StopWordsRemover(inputCol="Tokens", outputCol="Tokens sin stopwords")
stopWordsRemoved_df = remover.transform(tokenized)
```

TF-IDF: Una vez se tiene el conjunto de terminos a procesar, se tiene que obtener la frecuencia

de terminos para cada documento, y esta funcionalidad la proporciona *HashingTF* de Spark, que genera en un DataFrame nuevo las frecuencias de terminos de un conjunto de terminos inicial, posteriormente se procesa este DataFrame obtenido con la funcion de transformación dada por la clase *IDF* de Spark.

```
hashingTF = HashingTF (inputCol="Tokens sin stopwords", outputCol="rawFeatures", numFeatures=200000)
tfVectors = hashingTF.transform(stopWordsRemoved_df)


tfVectors.show()


idf = IDF(inputCol="rawFeatures", outputCol="features", minDocFreq=5)
idfModel = idf.fit(tfVectors)


tfIdfVectors = idfModel.transform(tfVectors)
```

K-Means: El paso final que se realizó en la implementación, es la utilización del algoritmo de agrupamiento k-means proporcionado por la libreria de Machine Learning de Spark el cual funciona a partir de los vectores TF-IDF obtenidos previamente, un k que es el número de agrupamientos, y un número de iteraciónes máximas.

La forma en la cual se implementó se puede ver en la siguiente ilustración.

```
# Trains a KMeans model.

kmeans = KMeans().setK(k).setMaxIter(maximoIter)

km_model = kmeans.fit(tfIdfVectors)


clustersTable = km_model.transform(tfIdfVectors)
```

5. Análisis de resultados

Se pone a ejecutar en el cluster con el dataset (en este caso `hdfs:///user/jestra52/small-gutenberg`) dado y un directorio donde se guardará el resultado (en este caso `hdfs:///user/jestra52/proyecto04_2`):

```
jestra52@hdplabmaster: [ ~/st0263-proyecto4 ]  (master)
$ spark-submit --master yarn --deploy-mode cluster --executor-memory 2G --num-executors 2 proyecto4.py hdfs:///user/jestra52/s
mall-gutenberg 4 10 hdfs:///user/jestra52/proyecto04_2
SPARK_MAJOR_VERSION is set to 2, using Spark2
17/11/22 23:14:42 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
17/11/22 23:14:43 WARN DomainSocketFactory: The short-circuit local reads feature cannot be used because libhadoop cannot be loaded.
17/11/22 23:14:43 INFO RMProxy: Connecting to ResourceManager at hdplabmaster/192.168.10.75:8050
17/11/22 23:14:43 INFO Client: Requesting a new application from cluster with 2 NodeManagers
17/11/22 23:14:43 INFO Client: Verifying our application has not requested more than the maximum memory capability of the cluster (57344 MB per container)
17/11/22 23:14:43 INFO Client: Will allocate AM container, with 1408 MB memory including 384 MB overhead
17/11/22 23:14:43 INFO Client: Setting up container launch context for our AM
17/11/22 23:14:43 INFO Client: Setting up the launch environment for our AM container
17/11/22 23:14:43 INFO Client: Preparing resources for our AM container
17/11/22 23:14:43 INFO HDFSCredentialProvider: getting token for namenode: hdfs://hdplabmaster:8020/user/jestra52
17/11/22 23:14:44 INFO DFSClient: Created HDFS_DELEGATION_TOKEN token 396715 for jestra52 on 192.168.10.75:8020
17/11/22 23:14:45 INFO RecoverableZooKeeper: Process identifier=hconnection-0x30c4e352 connecting to ZooKeeper ensemble=hdplabmaster:2181
17/11/22 23:14:45 INFO ZooKeeper: Client environment:zookeeper.version=3.4.6-129--1, built on 05/31/2017 02:32 GMT
17/11/22 23:14:45 INFO ZooKeeper: Client environment:host.name=hdplabmaster
17/11/22 23:14:45 INFO ZooKeeper: Client environment:java.version=1.8.0_144
17/11/22 23:14:45 INFO ZooKeeper: Client environment:java.vendor=Oracle Corporation
17/11/22 23:14:45 INFO ZooKeeper: Client environment:java.home=/opt/jdk8/jre
17/11/22 23:14:45 INFO ZooKeeper: Client environment:java.class.path=/usr/hdp/current/spark2-historyserver/conf/:/usr/hdp/current/spark2-client/jars/hk2-utils-2.4.0-b34.jar:/usr/hdp/current/spark2-client/jars/JavaEWAH-0.3.2.jar:/usr/hdp/current/spark2-client/jars/curator-framework-2.6.0.jar:/usr/hdp/current/spark2-client/jars/RoaringBitmap-0.5.11.jar:/usr/hdp/current/spark2-client/jars/httpclient-4.5.2.jar:/usr/hdp/current/spark2-client/jars/ST4-4.0.4.jar:/usr/hdp/current/spark2-client/jars/curator-recipes-2.6.0.jar:/usr/hdp/current/spark2-client/jars/activation-1.1.1.jar:/usr/hdp/current/spark2-client/jars/jsr305-1.3.9.jar:/usr/hdp/current/spark2-client/jars/antlr-2
```

Se puede ver que la ejecución fue aceptada y está corriendo:

```
17/11/22 23:14:48 INFO Client:
        client token: Token { kind: YARN_CLIENT_TOKEN, service:  }
        diagnostics: AM container is launched, waiting for AM container to Register with RM
        ApplicationMaster host: N/A
        ApplicationMaster RPC port: -1
        queue: default
        start time: 1511410487649
        final status: UNDEFINED
        tracking URL: http://hdplabmaster:8088/proxy/application_1511107733198_0321/
        user: jestra52
17/11/22 23:14:49 INFO Client: Application report for application_1511107733198_0321 (state: ACCEPTED)
17/11/22 23:14:50 INFO Client: Application report for application_1511107733198_0321 (state: ACCEPTED)
17/11/22 23:14:51 INFO Client: Application report for application_1511107733198_0321 (state: ACCEPTED)
17/11/22 23:14:52 INFO Client: Application report for application_1511107733198_0321 (state: ACCEPTED)
17/11/22 23:14:53 INFO Client: Application report for application_1511107733198_0321 (state: ACCEPTED)
17/11/22 23:14:54 INFO Client: Application report for application_1511107733198_0321 (state: ACCEPTED)
17/11/22 23:14:55 INFO Client: Application report for application_1511107733198_0321 (state: ACCEPTED)
17/11/22 23:14:56 INFO Client: Application report for application_1511107733198_0321 (state: ACCEPTED)
17/11/22 23:14:57 INFO Client: Application report for application_1511107733198_0321 (state: RUNNING)
17/11/22 23:14:57 INFO Client:
        client token: Token { kind: YARN_CLIENT_TOKEN, service:  }
        diagnostics: N/A
        ApplicationMaster host: 192.168.10.77
        ApplicationMaster RPC port: 0
        queue: default
        start time: 1511410487649
        final status: UNDEFINED
        tracking URL: http://hdplabmaster:8088/proxy/application_1511107733198_0321/
        user: jestra52
17/11/22 23:14:58 INFO Client: Application report for application_1511107733198_0321 (state: RUNNING)
17/11/22 23:14:59 INFO Client: Application report for application_1511107733198_0321 (state: RUNNING)
17/11/22 23:15:00 INFO Client: Application report for application_1511107733198_0321 (state: RUNNING)
17/11/22 23:15:01 INFO Client: Application report for application_1511107733198_0321 (state: RUNNING)
17/11/22 23:15:02 INFO Client: Application report for application_1511107733198_0321 (state: RUNNING)
17/11/22 23:15:03 INFO Client: Application report for application_1511107733198_0321 (state: RUNNING)
17/11/22 23:15:04 INFO Client: Application report for application_1511107733198_0321 (state: RUNNING)
17/11/22 23:15:05 INFO Client: Application report for application_1511107733198_0321 (state: RUNNING)
17/11/22 23:15:06 INFO Client: Application report for application_1511107733198_0321 (state: RUNNING)
17/11/22 23:15:07 INFO Client: Application report for application_1511107733198_0321 (state: RUNNING)
```

Y finaliza satisfactoriamente:

```
17/11/22 23:16:53 INFO Client: Application report for application_1511107733198_0321 (state: RUNNING)
17/11/22 23:16:54 INFO Client: Application report for application_1511107733198_0321 (state: RUNNING)
17/11/22 23:16:55 INFO Client: Application report for application_1511107733198_0321 (state: RUNNING)
17/11/22 23:16:56 INFO Client: Application report for application_1511107733198_0321 (state: RUNNING)
17/11/22 23:16:57 INFO Client: Application report for application_1511107733198_0321 (state: RUNNING)
17/11/22 23:16:58 INFO Client: Application report for application_1511107733198_0321 (state: RUNNING)
17/11/22 23:16:59 INFO Client: Application report for application_1511107733198_0321 (state: RUNNING)
17/11/22 23:17:00 INFO Client: Application report for application_1511107733198_0321 (state: RUNNING)
17/11/22 23:17:01 INFO Client: Application report for application_1511107733198_0321 (state: RUNNING)
17/11/22 23:17:02 INFO Client: Application report for application_1511107733198_0321 (state: RUNNING)
17/11/22 23:17:03 INFO Client: Application report for application_1511107733198_0321 (state: RUNNING)
17/11/22 23:17:04 INFO Client: Application report for application_1511107733198_0321 (state: RUNNING)
17/11/22 23:17:05 INFO Client: Application report for application_1511107733198_0321 (state: RUNNING)
17/11/22 23:17:06 INFO Client: Application report for application_1511107733198_0321 (state: RUNNING)
17/11/22 23:17:07 INFO Client: Application report for application_1511107733198_0321 (state: RUNNING)
17/11/22 23:17:08 INFO Client: Application report for application_1511107733198_0321 (state: RUNNING)
17/11/22 23:17:09 INFO Client: Application report for application_1511107733198_0321 (state: RUNNING)
17/11/22 23:17:10 INFO Client: Application report for application_1511107733198_0321 (state: FINISHED)
17/11/22 23:17:10 INFO Client:
        client token: Token { kind: YARN_CLIENT_TOKEN, service:  }
        diagnostics: N/A
        ApplicationMaster host: 192.168.10.77
        ApplicationMaster RPC port: 0
        queue: default
        start time: 1511410487649
        final status: SUCCEEDED
```

Se puede ver que se guardó el resultado en el el directorio `hdfs:///user/jestra52/proyecto04_2`:

```
jestra52@hdplabmaster: [ ~/st0263-proyecto4 ]  (master)
$ hdfs dfs -ls /user/jestra52/
Found 15 items
drwxrwxrwx   - jestra52 bigdata-estud          0 2017-11-19 13:00 /user/jestra52/.Trash
drwxr-xr-x   - jestra52 bigdata-estud          0 2017-11-22 23:17 /user/jestra52/.sparkStaging
drwx------   - jestra52 bigdata-estud          0 2017-11-09 19:10 /user/jestra52/.staging
drwxrwxrwx   - jestra52 bigdata-estud          0 2017-10-29 09:30 /user/jestra52/data_out1
drwxrwxrwx   - jestra52 bigdata-estud          0 2017-10-29 09:44 /user/jestra52/data_out2
drwxrwxrwx   - jestra52 bigdata-estud          0 2017-10-17 12:45 /user/jestra52/datasets
drwxrwxrwx   - jestra52 bigdata-estud          0 2017-11-09 18:50 /user/jestra52/jestra52_empl
drwxrwxrwx   - jestra52 bigdata-estud          0 2017-10-31 07:56 /user/jestra52/mrsalario
drwxrwxrwx   - jestra52 bigdata-estud          0 2017-11-09 19:00 /user/jestra52/mysqlOut
drwxr-xr-x   - jestra52 bigdata-estud          0 2017-11-09 19:09 /user/jestra52/mysqlin
drwxr-xr-x   - jestra52 bigdata-estud          0 2017-11-22 23:17 /user/jestra52/proyecto04_2
drwxr-xr-x   - jestra52 bigdata-estud          0 2017-11-22 19:25 /user/jestra52/proyecto4
drwxr-xr-x   - jestra52 bigdata-estud          0 2017-11-22 23:11 /user/jestra52/small-gutenbe
-rwxrwxrwx   2 jestra52 bigdata-estud        147 2017-10-31 07:54 /user/jestra52/test.txt
drwxrwxrwx   - jestra52 bigdata-estud          0 2017-10-29 09:35 /user/jestra52/tmp
```

Finalmente imprimimos el resultado y esto es lo que obtenemos:

```
jestra52@hdplabmaster: [ ~/st0263-proyecto4 ]  (master)
$ hdfs dfs -ls /user/jestra52/proyecto04_2
Found 2 items
-rw-r--r--   2 jestra52 bigdata-estud          0 2017-11-22 23:17 /user/j
-rw-r--r--   2 jestra52 bigdata-estud       6081 2017-11-22 23:17 /user/j
0-6d9d-4a66-a937-786d17f31a06.csv
jestra52@hdplabmaster: [ ~/st0263-proyecto4 ]  (master)
$ hdfs dfs -cat /user/jestra52/proyecto04_2/part-00000-6f1fea80-6d9d-4a66
Ruta,prediction
hdfs://hdplabmaster:8020/user/jestra52/small-gutenberg/45250-8.txt,1
hdfs://hdplabmaster:8020/user/jestra52/small-gutenberg/45438-8.txt,1
hdfs://hdplabmaster:8020/user/jestra52/small-gutenberg/45508-8.txt,0
hdfs://hdplabmaster:8020/user/jestra52/small-gutenberg/45770-8.txt,1
hdfs://hdplabmaster:8020/user/jestra52/small-gutenberg/45829-8.txt,0
hdfs://hdplabmaster:8020/user/jestra52/small-gutenberg/45830-8.txt,0
hdfs://hdplabmaster:8020/user/jestra52/small-gutenberg/45831-8.txt,0
hdfs://hdplabmaster:8020/user/jestra52/small-gutenberg/45832-8.txt,0
hdfs://hdplabmaster:8020/user/jestra52/small-gutenberg/45833-8.txt,0
hdfs://hdplabmaster:8020/user/jestra52/small-gutenberg/45834-8.txt,0
hdfs://hdplabmaster:8020/user/jestra52/small-gutenberg/45835-8.txt,0
hdfs://hdplabmaster:8020/user/jestra52/small-gutenberg/45836-8.txt,0
hdfs://hdplabmaster:8020/user/jestra52/small-gutenberg/45837-8.txt,0
hdfs://hdplabmaster:8020/user/jestra52/small-gutenberg/45945-8.txt,0
hdfs://hdplabmaster:8020/user/jestra52/small-gutenberg/46000-8.txt,0
hdfs://hdplabmaster:8020/user/jestra52/small-gutenberg/46182-8.txt,0
hdfs://hdplabmaster:8020/user/jestra52/small-gutenberg/46196-8.txt,0
hdfs://hdplabmaster:8020/user/jestra52/small-gutenberg/46201-8.txt,2
hdfs://hdplabmaster:8020/user/jestra52/small-gutenberg/46246-0.txt,3
hdfs://hdplabmaster:8020/user/jestra52/small-gutenberg/46246-8.txt,1
```

En el resultado se puede ver el nombre archivo y, separado por una coma, el número del cluster (que nos dio el K-means) al que pertenece el archivo.

6. Conclusiones

- Durante el desarrollo de ésta práctica se obtuvieron nuevos conocimientos sobre Apache Spark, que es un comienzo muy completo para tener una introducción al mundo de BigData.
- Se logra tener un acercamiento a los aspectos básicos en cuanto a la programación en BigData, analizando y comprendiendo sus funcionalidades, ventajas y desventajas.
- Se entendió el proceso de ETL al momento de realizar el diseño y análisis de algoritmos distribuidos

.

Referencias

[1] A. Huang. *Similarity measures for text document clustering*. The University of Waikato, Hamilton, New Zealand. 2008.

[2] A. Trevino. *Introduction to K-means clustering*. Fuente: `www.datascience.com/blog/k-means-clustering`. 2016.

[3] Ming Chen. *TF-IDF, HashingTF and CountVectorizer*. Fuente: `https://mingchen0919.github.io/learning-apache-spark/tf-idf.html`. 2015.

[4] Timothy Fox. *Clustering with K-Means with Spark and MLlib*. Fuente:

http://timothyfox.net/?p=15.
2017.

[5] logicalguess (usuario de Github). *TF-IDF with Spark for the Kaggle popcorn competition*.
Fuente:
https://github.com/logicalguess/tf-idf-spark-and-python.
2015.

[6] Spark org. *Feature Extraction and Transformation - RDD-based API - Spark 2.2.0 Documentation*.
Fuente:
https://spark.apache.org/docs/2.2.0/mllib-feature-extraction.html.
2015.

[7] Wikipedia. *Extract, transform and load*
Fuente:
https://es.wikipedia.org/wiki/Extract,_transform_and_load.