




MÁSTER

INFRAESTRUCTURAS COMPUTACIONALES PARA EL
PROCESAMIENTO DE DATOS MASIVOS

TRABAJO PRÁCTICO MÓDULO 3
GESTIÓN DE DATOS EN TIEMPO REAL (*STREAMING*)



2021-2022

Dr. Andrés Duque Fernández – Dr. Rafael Pastor Vargas

Dr. Agustín C. Caminero Herráez – Dr. Salvador Ros Muñoz

MÁSTER UNIVERSITARIO EN INGENIERÍA Y CIENCIA DE DATOS

TRABAJO PRÁCTICO MÓDULO 3:

KAFKA + SPARK STREAMING

CONSIDERACIONES INICIALES:

- En este documento se presenta el Trabajo Práctico (TP) del módulo 3 de la asignatura “INFRAESTRUCTURAS COMPUTACIONALES PARA EL PROCESAMIENTO DE DATOS MASIVOS”, del “MÁSTER UNIVERSITARIO EN INGENIERÍA Y CIENCIA DE DATOS” de la UNED.
- Este trabajo se realiza de forma individual. En las siguientes secciones se exponen los diferentes ejercicios que es necesario implementar para este trabajo.
- Se recomienda utilizar el Sistema Operativo Linux, en concreto la distribución Ubuntu.
- Descargar la última versión de Apache Kafka desde <https://kafka.apache.org> y descomprimirla en la carpeta local.
- Los archivos de configuración de Kafka se encuentran dentro de la carpeta “config”, y los scripts para ejecutar los distintos comandos dentro de la carpeta “bin”.
- Spark deberá estar instalado y configurado tal y como se ha realizado en el módulo 2 de la asignatura. Se deberá tener localizado el PATH de instalación de Spark ya que será necesario indicarlo en el notebook Jupyter.
- Se utilizará el lenguaje de programación Python 3 y en concreto la API que proporciona para trabajar sobre Spark (pyspark).
- Desarrollar los scripts de python utilizando notebooks de Jupyter, para que sea más sencillo entregar, además del código desarrollado, cualquier comentario que el estudiante quiera añadir.
- Se recomienda haber ejecutado los notebooks de ejemplo que se proporcionan en el módulo para afianzar conocimientos sobre la inicialización del contexto Spark y Spark Streaming, así como la inicialización y terminación de aplicaciones Spark Streaming.

- Se recomienda especialmente la visualización del vídeo dedicado a Spark Streaming sobre Kafka de la colección de vídeos recomendada como material adicional del tema 3 del módulo:
<https://learning.oreilly.com/videos/apache-spark-streaming/9781789808223> (el vídeo en concreto se denomina PySpark Streaming with Apache Kafka).
- Los paquetes de python que se van a utilizar, y que habrá que instalar utilizando pip en caso de que no estén disponibles son: pyspark, findspark y kafka-python.
- Para los ejercicios del apartado 2, se recomienda inicializar el contexto Spark en local, con tantos *threads* como núcleos lógicos tenga la máquina sobre la que se ejecuta:

```
sc = SparkContext("local[*]").
```
- Como siempre, para cualquier duda o comentario el contacto con el equipo docente será a través de los foros de la asignatura.
- **ES NECESARIO REALIZAR LOS DOS APARTADOS PARA APROBAR ESTE TRABAJO (ES DECIR, AUNQUE EL APARTADO 2 VALGA 6 PUNTOS SI NO SE ENTREGAN RESPUESTAS PARA EL APARTADO 1 NO SE APROBARÁ EL TRABAJO).**

APARTADO 1: KAFKA (4 puntos)

EJERCICIOS

1. Localizar el archivo que contiene la configuración del servidor Zookeeper y levantar dicho servidor utilizando la configuración por defecto. ¿Qué puerto es el utilizado por defecto por Zookeeper?
2. Localizar el archivo que contiene la configuración de un broker y levantar dicho broker. ¿Qué puerto es el utilizado por defecto por los brokers?
3. Crear un topic nuevo con el nombre "initial", con una partición y un factor de replicación igual a 1.
4. Describir el topic "initial" y comentar los resultados que se obtienen.
5. Arrancar un productor sobre el topic "initial".
6. Arrancar un consumidor sobre el topic "initial" que consuma todos los mensajes desde el principio.
7. Generar algunos mensajes en el productor levantado en el ejercicio 5 y observar cómo se reciben en el consumidor levantado en el ejercicio 6. Comentar la salida obtenida. Detener a continuación el productor y el consumidor.
8. Crear un topic nuevo con el nombre "testRep", con dos particiones y un factor de replicación igual a 2. ¿Qué sucede? ¿Cuál es la razón?
9. Detener el broker levantado en el ejercicio 2.
10. Levantar tres brokers distintos: generar nuevos ficheros de configuración para cada uno de ellos indicando las modificaciones que han sido necesarias. Los identificadores de los brokers serán 0, 1 y 2.
11. Describir el topic "initial" comentar los resultados que se obtienen.

12. Crear un topic nuevo con el nombre “testOrder1”, con 1 partición y factor de replicación igual a 3.
13. Crear un topic nuevo con el nombre “testOrder2”, con 3 particiones y factor de replicación igual a 3.
14. Describir los topics “testOrder1” y “testOrder2” y comentar los resultados que se obtienen.
15. Arrancar un productor sobre el topic “testOrder1” y un consumidor desde el inicio sobre el mismo topic. Introducir algunos mensajes y observar cómo se consumen. Detener el productor y el consumidor.
16. Arrancar un productor sobre el topic “testOrder2” y un consumidor desde el inicio sobre el mismo topic. Introducir algunos mensajes y observar cómo se consumen. Detener el productor y el consumidor.
17. Arrancar un consumidor sobre el topic “testOrder1”, sin que consuma desde el principio, únicamente los valores que le lleguen desde el nuevo arranque. En otra consola, generar una secuencia de números del 1 al 30 (utilizar el comando “seq”) y enviársela mediante una tubería o pipe (indicada con el carácter “|”) a un productor sobre el topic “testOrder1”. Observar cómo se consume dicha secuencia en el consumidor.
18. Arrancar un consumidor sobre el topic “testOrder2”, sin que consuma desde el principio, únicamente los valores que le lleguen desde el nuevo arranque. En otra consola, generar una secuencia de números del 1 al 30 (utilizar el comando “seq”) y enviársela mediante una tubería o pipe (indicada con el carácter “|”) a un productor sobre el topic “testOrder2”. Observar cómo se consume dicha secuencia en el consumidor.
19. ¿Cuáles son las diferencias entre lo mostrado en el consumidor en los dos ejercicios anteriores? ¿A qué se deben?

20. Obtener los últimos offsets de los mensajes del topic “testOrder1” y “testOrder2”. ¿Qué se observa? (Utilizar el comando `kafka-run-class` con la opción `kafka.tools.GetOffsetShell`).
21. Detener de forma no planificada (utilizando Ctrl-C o cerrando la consola) el broker con identificador 2, y describir el topic “testOrder2”. ¿Cuáles son las diferencias con la descripción que obtuvimos en el ejercicio 14? ¿A qué se deben?

APARTADO 2: KAFKA + SPARK STREAMING (6 puntos)

El análisis en tiempo real de redes sociales como Twitter es una de las grandes tendencias en el campo del *big data* en la actualidad. El seguimiento de eventos a través de sus hashtags, el análisis de tendencias de opinión o la gestión de la reputación de empresas son sólo algunos de los casos de uso relacionados con el *streaming* de datos de Twitter. Aunque el acceso programático a tweets se puede realizar utilizando la API de Twitter Developer (developer.twitter.com), en esta práctica vamos a simular su funcionamiento para evitar las gestiones que se necesitan para obtener las credenciales necesarias para dicha API.

Para ello, se utilizará un dataset previamente descargado y disponible públicamente, que se proporcionará en forma de archivo de texto, “sentweets_esp.txt”. Con este origen de datos, se utilizarán las dos tecnologías estudiadas en el presente módulo (Kafka y Spark Streaming) para la transmisión de los datos y para su procesamiento, respectivamente.

El primer paso a realizar es la configuración del sistema basado en Kafka que se va a utilizar: Se deberá levantar el servidor zookeeper, así como un clúster de 2 brokers. A continuación, se creará un *topic* con el nombre **kafkaTwitter**, con dos particiones y un factor de replicación igual a 2.

Se proporciona como recurso un script de python que lee del fichero “sentweets_esp.txt” los tweets y los vuelca al *topic* kafkaTwitter creado anteriormente, utilizando un productor, a través de la API de Kafka para Python (si no está instalada ejecutar “pip install kafka-python”). Para simular el funcionamiento de un sistema de *streaming*, se aleatorizan los tweets y se vuelca cada línea del fichero (que representa un tweet) al sistema Kafka dejando un tiempo aleatorio entre 0 y 0.1 segundos entre dos tweets. **No hace falta ejecutar este script dentro de un notebook de Jupyter, ya que se trata de un script “.py” sencillo, por lo que se puede ejecutar simplemente mediante “python3 tweet_producer.py”.** Es decir, una vez que se levante el cluster Kafka, al ejecutar este script se estarán volcando tweets al *topic* de forma continuada. El script se puede detener y volver a ejecutar cuando se desee.

Los recursos proporcionados son los siguientes:

- Fichero “**sentweets_esp.txt**”, que contiene la colección completa de tweets a utilizar.
- Script “**tweet_producer.py**”, que lee todos los tweets, los aleatoriza y los vuelca al topic de Kafka.
- Fichero “**positive_lex.txt**”, sólo para el ejercicio 3.
- Fichero “**negative_lex.txt**”, sólo para el ejercicio 3.

Nótese que el fichero “**tweet_producer.py**” y “**sentweets_esp.txt**” han de estar almacenados en la misma carpeta para el correcto funcionamiento del script de python.

Se recomienda inicializar el contexto Spark en local, con tantos *threads* como núcleos lógicos tenga la máquina sobre la que se ejecuta: `sc = SparkContext("local[*]")`.

EJERCICIOS

Ejercicio 1 (conteo de tweets): (1 punto)

Desarrollar un notebook de Jupyter, denominado “**tweetCount.ipynb**” en el que se utilice como fuente de datos Kafka, y en concreto el *topic* `kafkaTwitter`, se establezca una duración de *batch* de un segundo, y se muestre, **cada 5 segundos**, el número de tweets recibidos **en los últimos 10 segundos**. ¿Alrededor de qué número (aproximado) se estabiliza el número de tweets que se procesan en el lapso de tiempo indicado (5 segundos)? ¿Tiene sentido? ¿Por qué?

Ejercicio 2 (hashtags más utilizados): (2 puntos)

Desarrollar un notebook de Jupyter, denominado “**hashtags.ipynb**”, en el que se utilice como fuente de datos Kafka, y en concreto el *topic* `kafkaTwitter`. La duración del *batch* será de 5 segundos. Se procesarán los tweets que lleguen para extraer los hashtags que contengan **(tener en cuenta que todos los hashtags comienzan por el carácter ‘#’)**. Se irán mostrando,

cada vez que se procese el batch (5 segundos) los diez hashtags más utilizados desde el inicio del programa hasta ese momento y el número total de apariciones de cada uno, ordenados de mayor a menor frecuencia.

Ejercicio 3 (análisis de sentimiento): (3 puntos)

Una de las aplicaciones más interesantes que se pueden realizar sobre la información que se recopila de Twitter es el denominado análisis de sentimiento o *sentiment analysis*. Se trata fundamentalmente de clasificar de forma automática la polaridad de un tweet en concreto, en función de su contenido, categorizándolo como positivo, negativo o neutro, e incluso asignándole un valor numérico a dicha clasificación. Esta funcionalidad se utiliza, por ejemplo, en sistemas de análisis de reputación online (*Online Reputation Management*, ORM) para determinar qué se está diciendo de una determinada marca en Twitter, en tiempo real.

En este ejercicio vamos a simular, de forma simplificada, un sistema de tiempo real que ofrezca una puntuación de polaridad (positiva, negativa o neutra) a los tweets que vayan llegando. Para ello, el modelo de clasificación se basará en listas de palabras, denominadas lexicones. En concreto, dispondremos de un lexicón de palabras positivas y su puntuación, “positive_lex.txt”, y de un lexicón de palabras negativas y su puntuación, “negative_lex.txt”. Ambos contienen, en cada línea, una palabra (positiva o negativa, respectivamente), seguida de su puntuación. **Nótese que la polaridad negativa ya está expresada con un signo menos.**

Se pide desarrollar un notebook de Jupyter, denominado “sentiment.ipynb”, en el que se utilice como fuente de datos Kafka, y en concreto el *topic* kafkaTwitter. Para cada nuevo tweet que llegue al sistema, se analizará su polaridad dividiendo el tweet por palabras y realizando la suma de la polaridad de todas las palabras que aparezcan en los ficheros “positive_lex.txt” y “negative_lex.txt”. **Las palabras del lexicon negativo tienen un signo menos delante, por lo que restarán a la puntuación final del tweet.** Si ninguna de las palabras del tweet está almacenada en los ficheros proporcionados (o si las puntuaciones de las palabras positivas y negativas se anulan), la polaridad del tweet será 0.

La salida del sistema ha de imprimirse cada segundo, y consistirá en una línea por cada tweet recibido, que contenga el texto del propio tweet, a continuación la puntuación de su polaridad, y la palabra “NEUTRO” si la puntuación es igual 0, “POSITIVO” si es mayor que 0, y “NEGATIVO” si es menor que 0.

Por ejemplo, tenemos el siguiente tweet:

'Hoy vuelvo a España, me ha encantado Polonia :D'

El sistema devolverá el texto del tweet, seguido de la puntuación 0.417, y de la palabra “POSITIVO”, ya que la palabra “encantado” está en el fichero “positive_lex.txt” con esa puntuación, y el resto de las palabras no se encuentran en ninguno de los dos lexicones.

Se recomienda modificar el ritmo de volcado de tweets a Kafka, para que la salida sea más legible. Para ello, modificar la línea 24 del script “tweet_producer.py” de **sleep(random.random()/10)** a **sleep(1)**.

DOCUMENTACIÓN A ENTREGAR

Se entregará un archivo comprimido en zip denominado “TP3-ApellidosNombre.zip”, donde *Apellidos* y *Nombre* han de sustituirse por los valores correspondientes de/de la estudiante. Al descomprimir, dentro de la carpeta raíz deberá haber dos carpetas, denominadas Apartado1 y Apartado2. Los contenidos de cada carpeta serán los siguientes:

Apartado 1:

Un documento en .doc o .pdf que contenga las respuestas a los ejercicios del apartado 1, ya sean los comandos utilizados en cada ejercicio, o las respuestas a las preguntas específicas de cada ejercicio.

Se valorará positivamente un comentario final sobre las impresiones del estudiante acerca de este apartado, así como las dificultades encontradas.

Apartado 2:

Un notebook de Jupyter para cada uno de los ejercicios propuestos:

- Ejercicio 1: Fichero countTweets.ipynb
- Ejercicio 2: Fichero hashtags.ipynb
- Ejercicio 3: Fichero sentiment.ipynb

Los notebooks deberán contener el código realizado, listo para ser ejecutado en el entorno de desarrollo. Se debe incluir no solamente el código sino también las explicaciones necesarias sobre cómo se han resuelto los ejercicios, respuestas a las preguntas planteadas en el enunciado, problemas y dificultades encontrados, etc., de tal forma que el notebook sea autocontenido. Al principio del notebook se debe indicar el nombre del/de la estudiante.

De forma optativa, se puede incluir una memoria explicativa en doc o pdf, que aporte contenido significativo adicional (referencias, documentos adicionales, reflexiones, etc.).

Al igual que en el apartado 2, se valorará positivamente que en la documentación (ya sea en los propios notebooks o en un documento aparte) se incluyan comentarios acerca de las dificultades encontradas y opiniones sobre la realización de este apartado, así como sobre la práctica en global y sobre el módulo 3 en general.

Importante: La puntuación de cada apartado no se refiere únicamente al correcto funcionamiento de la solución entregada, sino también a la existencia de una documentación lo suficientemente completa, correctamente escrita y explicativa de todos los aspectos que el estudiante considere importantes (decisiones sobre el código, dificultades, opiniones, etc.).