DOCUMENTACIÓN PRÁCTICA 3- SPARK

Documentación sobre la tercera Práctica de Grandes Volúmenes de Datos

DANIEL SABBAGH
4º ING. INFORMÁTICA

INDICE

1. INTRODUCCIÓN Y OBJETIVOS

Se tendrá un conjunto de datos compuesto por 3 archivos CSV:

- Archivo con productos de Amazon, Amazon.csv
- Archivo con productos de Google, GoogleProducts.csv
- Archivo con los resultados de la comparativa de los dos archivos anteriores, Amazon_GoogleProducts_perfectMaping.csv

En la práctica se usarán activamente los dos primeros archivos, estos archivos contienen productos con 5 atributos: Su id, el titulo o nombre, la descripción, el fabricante y el precio.

Se propone implementar un algoritmo en PySpark llevando a cabo una resolución de entidades. Dado que son dos archivos aparentemente diferentes, se busca que, mediante el algoritmo implementado, se encuentren productos similares en ambos catálogos.

2. RESOLUCIÓN

Para la resolución del problema se han seguido las siguientes fases:

Normalización de fuentes

La normalización de fuentes en PySpark es el proceso de estructurar y organizar los datos en un formato consistente y estandarizado para su análisis.

Se describirá por pasos:

- Paso 1: Unir los dos csv mediante un dataframe, primeramente, cargamos cada catálogo csv en un dataframe diferente, para su posterior unión. En la unión es necesario que los headers de ambos datasets sean iguales, por lo que es necesaria el renombramiento de la columna “title” del csv de amazon, cambiarla a “name”. Tras esto, se unen los dos dataframes en uno.
- Paso 2: Se busca convertir los datos en algo manejable, por lo que, se pasarán varias columnas del dataframe a minúsculas junto con deshacerse de los espacios a los lados
- Paso 3: Una vez terminado el paso 2, se descartarán las filas del dataframe con valores nulos y las filas duplicadas.

Tokenización

Proceso por el cual el texto de las columnas elegidas del dataframe se convertirán en una lista de todas las palabras de la celda, cada palabra se denomina token. Se retornará al módulo main.py el resultado de la normalización

Pares de Candidatos

Para la resolución de pares de candidatos se usa la función HasingTF, la cual convierte una lista de palabras en un vector de características. Una vez generada, cuenta el número de veces que se repite cada palabra y las agrupa.

Posteriormente la función MinHashLSH, reduce los datos para que se pueda hacer una búsqueda de similitud eficiente, explicado en el siguiente apartado de la documentación.

Se devuelven los datos al fichero main.py para la fase de puntuación.

Puntuar Pares de Candidatos

Para puntuar los pares de candidatos se usa una métrica llamada índice de Jaccard, ésta se utiliza para comparar la similitud entre conjuntos de elementos. Se calcula dividiendo el número de elementos comunes entre los conjuntos, por el número total de elementos en los conjuntos.

Una vez realizado, se hará un filter para que me devuelva únicamente los productos similares con más de un 30% de índice de similitud.

3. CONCLUSIÓN

Al realizar la práctica he podido ver que PySpark es una herramienta poderosa para el análisis y procesamiento de grandes conjuntos de datos. La comparación de catálogos de datos es una tarea crítica en entornos muchos empresariales y la utilización de PySpark ha permitido una mayor eficiencia y precisión que si utilizara otra herramienta como MapReduce.

4. BIBLIOGRAFÍA

- https://spark.apache.org/docs/latest/api/python/
- https://spark.apache.org/docs/latest/sql-getting-started.html
- https://spark.apache.org/docs/latest/sql-programming-guide.html
- https://medium.com/@crscardellino/procesando-datos-con-spark-y-iv-corriendo-una-aplicaci%C3%B3n-con-pyspark-5c26e828465d
- https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.ml.feature.HashingTF.html
- https://en.wikipedia.org/wiki/Locality-sensitive_hashing
- https://spark.apache.org/docs/3.1.1/api/python/reference/api/pyspark.ml.feature.MinHashLSH.html
- https://stackoverflow.com/questions/52923110/spark-python-how-to-calculate-jaccard-similarity-between-each-line-within-an-rd
- https://www.statology.org/jaccard-similarity-python/