

**EKONOMICKÁ UNIVERZITA V BRATISLAVE
FAKULTA HOSPODÁRSKEJ INFORMATIKY**

**Analýza neštruktúrovaného a semištruktúrovaného
datasetu pomocou Sparku**

Zadanie z predmetu Big Data

2023

Bc. Ondrej Šima

Bc. Alena Stracenská

Obsah

Úvod	4
1 Analýza semištrukturovaného datasetu	5
1.1 Použité dáta	5
1.2 Tvorba <code>sparkapp.py</code> skriptu	5
Záver	9
Zoznam použitej literatúry	10

Zoznam obrázkov a tabuliek

Obrázok 1	Ukážka súboru s logmi, zdroj: [vlastné spracovanie]	5
Obrázok 2	Spustenie <code>sparkapp.py</code> skriptu, zdroj: [vlastné spracovanie]	7
Obrázok 3	Ukážka výsledných <code>.csv</code> súborov, zdroj: [vlastné spracovanie] . . .	7
Obrázok 4	Obsah <code>.csv</code> súboru, zdroj: [vlastné spracovanie]	8

Úvod

Spark bol vyvinutý na Univerzite Kalifornie v Berkeley ako náhrada za Hadoop MapReduce a ponúka niekoľko výhod, ako napríklad rýchlejšie spracovanie dát, podpora množstva dátových zdrojov a možnosť spracovania úloh v reálnom čase.

Pyspark je Python rozhranie pre Spark, ktoré umožňuje používanie Sparku s Pythonom. Je ho vhodné používať pre spracovanie veľkých objemov dát a paralelizáciu úloh, ktoré by boli ťažko spracovateľné na jednom stroji. Umožňuje spracovanie rôznych typov dát, ako sú štruktúrované, neštruktúrované alebo pološtruktúrované dáta.

V tomto zadaní si napíšeme vlastný `sparkapp.py` skript, ktorý si prispôbime podľa toho, aký výstup chceme dosiahnuť. Následne si ho zapíšeme do výsledného súboru, v ktorom si správnosť spracovania logov overíme.

Veríme, že tento dokument bude prínosný pre čitateľa nie len po teoretickej stránke, ale aj po praktickej, kde sa naučí ako možno spracovať dáta pomocou Sparku resp. Pysparku.

1 Analýza semištruktúrovaného datasetu

1.1 Použité dáta

Dáta, s ktorými sme pracovali, nie len teraz, ale aj v predošlých zadaniach, boli vo forme apačovských webových logov. Môžeme si ich stiahnuť na kaggle.com. Sú to pološtruktúrované dáta, ktorých veľkosť obmedzíme na 100 000 riadkov.



Obrázok 1: Ukážka súboru s logmi, zdroj: [vlastné spracovanie]

1.2 Tvorba sparkapp.py skriptu

Najprv sme stiahli `spark-3.3.2-bin-hadoop3.tgz` pre nami použitú verziu Hadoopu. Po rozbalení a nastavení systémovej premennej `PATH` sme nainštalovali PySpark cez [1]:

```
$ pip install pyspark
$ pyspark
```

V zdrojovom kóde sme najskôr nainportovali potrebné knižnice, následne nastavili Spark reláciu. Súbor budeme načítavať do Spark DataFrame. Najskôr sme pre súbor vytvorili schému, čiže priradili jednotlivým poliam logu ktorý budeme parsovať, názov a typ. Načítaný log zo súboru potom parsujeme do jednotlivých polí DataFrame pomocou

regexov. Spracovaný log sme potom uložili ako štrukturovaný .csv súbor a zobrazili. Kód v skriptesparkapp.py je nasledovný:

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import split, regexp_extract
from pyspark.sql.types import StructType, StructField, StringType, IntegerType

# Spark relacia
spark = SparkSession.builder.appName("LogParser").getOrCreate()

#cesta k suboru
log_file_path = "/home/alena/access_100K.log"

# nastav schema
schema = StructType([
    StructField("ip", StringType(), True),
    StructField("date_time", StringType(), True), #zatiaľ ako string
    StructField("request", StringType(), True),
    StructField("status", IntegerType(), True),
    StructField("bytes_sent", IntegerType(), True),
    StructField("referrer", StringType(), True),
    StructField("user_agent", StringType(), True),
    StructField("other", StringType(), True)])

#nacitanie zo suboru
log_df = spark.read.text(log_file_path)

#parsovanie do poli
parsed_df = log_df.select(
    regexp_extract('value', r'^([\s]+)', 1).alias('ip'),
    regexp_extract('value', r'\[(.*)\]', 1).alias('date_time'),
    regexp_extract('value', r'"([\^"]+)"', 1).alias('request'),
    regexp_extract('value', r'\s(\d{3})\s', 1).cast(IntegerType()).alias('status'),
    regexp_extract('value', r'\s(\d+) \s"', 1).cast(IntegerType()).alias('bytes_sent'),
    regexp_extract('value', r'"([\^"]+)"\s+"([\^"]*)"', 1).alias('referrer'),
    regexp_extract('value', r'"([\^"]+)"\s+"([\^"]*)"', 1).alias('user_agent'),
    regexp_extract('value', r'\s+"([\^"]+)"$', 0).alias('other')
).drop("other")

parsed_df.write.csv("/home/alena/parsed_logs.csv")
parsed_df.show(truncate=False)
```

Následne sme spustili daný skript pomocou príkazu:

```
python3 sparkapp.py
```

Výstupom programu je náhľad (prvých 20 riadkov) z vyparovaného súboru a vyparovaný, štrukturovaný `.csv` súbor.

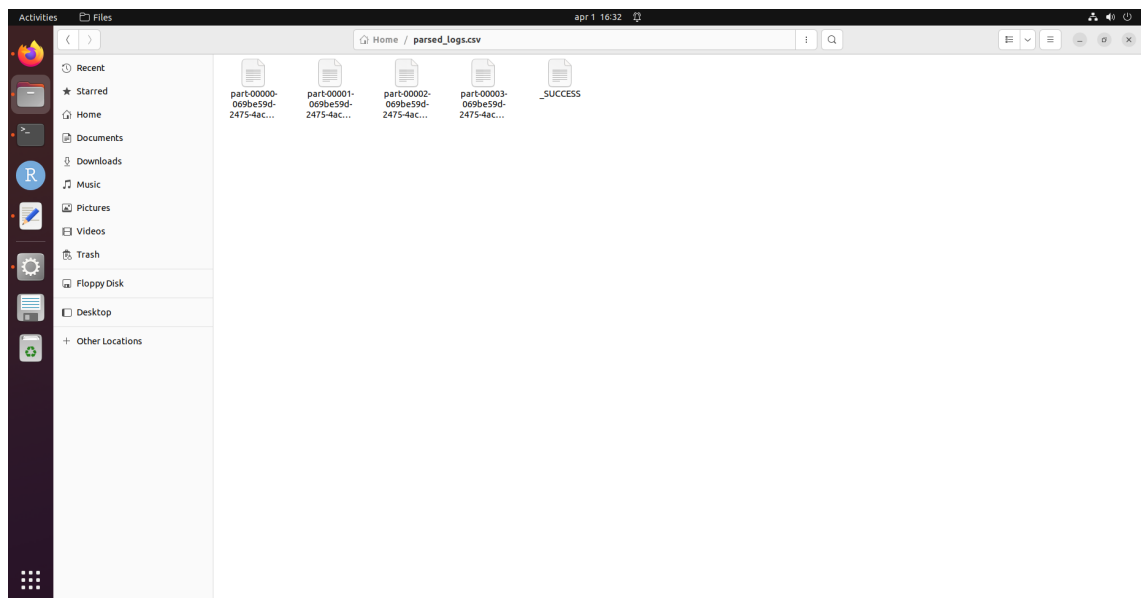
The screenshot shows a terminal window titled "alena@alena-virtual-machine: ~". The user has executed the command `$ python3 sparkapp.py`. The output consists of several lines of log messages:

- `23/04/01 16:31:23 WARN Utils: Your hostname, alena-virtual-machine resolves to a loopback address: 127.0.0.1; using 192.168.109.128 instead (on interface ens33)`
- `23/04/01 16:31:23 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address`
- `Setting default log level to "WARN".`
- `To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).`
- `23/04/01 16:31:24 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable`

Below the logs, there is a table-like representation of HTTP requests being processed. Each row contains fields such as IP, date_time, request, status, bytes_sent, referrer, and user_agent.

ip	date_time	request	status	bytes_sent	referrer	user_agent
154.36.149.41	[22/Jan/2019:03:56:14 +0330]	GET /filter/2713n20N0YK5NDANAF7XDBA7XD9XBENBNCMDANA9NDBN3ND984.271NDANAYK9XK5NBDAANBN1NZDN0NKA7XDBN8DZ205N20ND9XK5XDAFA7XDBNA7XD9XBENBNCMDANA9NDBN3ND984.p53 HTTP/1.1	200	10577	-	zillya/5.0 (compatible; AhrefsBot/6.1; +http://ahrefs.com/robot/)
131.56.96.51	[22/Jan/2019:03:56:16 +0330]	GET /vimage/60844/productModel/200x200 HTTP/1.1	200	15667	https://www.zanbil.ir/n/filiter/b113	zillya/5.0 (Linux; Android 6.0; ALE-L21 Build/HuaweiALE-L21; AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.158 Mobile Safari/537.36
131.56.96.51	[22/Jan/2019:03:56:16 +0330]	GET /vimage/61474/productModel/200x200 HTTP/1.1	200	15379	https://www.zanbil.ir/n/filiter/b113	zillya/5.0 (Linux; Android 6.0; ALE-L21 Build/HuaweiALE-L21; AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.158 Mobile Safari/537.36
140.77.167.129	[22/Jan/2019:03:56:17 +0330]	GET /vimage/14925/productModel/100x100 HTTP/1.1	200	1696	-	zillya/5.0 (compatible; bingbot/2.0; +http://www.bing.com/bingbot.htm)
191.99.72.15	[22/Jan/2019:03:56:17 +0330]	GET /product/31893/62100/XDBN83NDBN84ND9XBENBNCMDANA9NDBN3ND984.271NDANAYK9XK5NBDAANBN1NZDN0NKA7XDBN8DZ205N20ND9XK5XDAFA7XDBNA7XD9XBENBNCMDANA9NDBN3ND984.p53 HTTP/1.1	200	14183	-	zillya/5.0 (Windows NT 6.2; Win64; x64; rv:16.0) Gecko/16.0 Firefox/16.0
140.77.167.129	[22/Jan/2019:03:56:17 +0330]	GET /vimage/23488/productModel/150x150 HTTP/1.1	200	12654	-	zillya/5.0 (compatible; bingbot/2.0; +http://www.bing.com/bingbot.htm)
140.77.167.129	[22/Jan/2019:03:56:18 +0330]	GET /vimage/15437/productModel/150x150 HTTP/1.1	200	13688	-	zillya/5.0 (compatible; bingbot/2.0; +http://www.bing.com/bingbot.htm)
140.77.167.129	[22/Jan/2019:03:56:18 +0330]	GET /vimage/576/article/100x100 HTTP/1.1	200	12654	-	zillya/5.0 (compatible; bingbot/2.0; +http://www.bing.com/bingbot.htm)

Obrázok 2: Spustenie `sparkapp.py` skriptu, zdroj: [vlastné spracovanie]



Obrázok 3: Ukážka výsledných .csv súborov, zdroj: [vlastné spracovanie]

Záver

Cieľom tohto zadania bolo opísať tvorbu `sparkapp.py` skriptu a logiku jeho vykonávania. Následne ho spustiť a overiť, či výstup, ktorý sme chceli bol dosiahnutý. Cieľ tohto zadania sme splnili.

Problémy sa nám vyskytli najmä pri tvorbe skriptu, kde nám trvalo dlhší čas pochopiť, ako sa pracuje so Spark DataFrame, ktorý sa líši od klasického Pandas DataFrame. Môžeme tak povedať, že logy sme spracovali a rozparsovali správne podľa jednotlivých kategórií a týmto deklarovali funkčnosť uvedeného Python kódu v skripte.

Veríme, že toto zadanie prinesie čitateľovi nový pohľad na možnosť spracovávať veľké súbory a získa nové prípadne obohatí už doterajšie vedomosti a schopnosti o tvorbu takýchto skriptov.

Zoznam použitej literatúry

- [1] APACHE SOFTWARE FOUNDATION. 2023. *Download Apache SparkTM* In spark.apache.org [online]. 2023. [citované dňa 1.4.2023]. Dostupné na internete: <https://spark.apache.org/downloads.html>.