BDA, Praktikumsbericht 3

Gruppe mi6xc: Alexander Kniesz, Maximilian Neudert, Oskar Rudolf

Quellen

Das PySpark Notebook findet man hier.

Aufgabe 1

Wir lesen das `txt` file als DataFrame ein. Anschließend kann man mit `withColumn` diverse Operationen auf den Spalten ausführen. Das DataFrame enthält alle Zeilen des `txt` Files als Zeilen. In unserem Fall haben wir die Daten zuerst bereinigt, sprich Sonderzeichen und leere Zeilen entfernt. Anschließend haben wir mittels `split` die Strings in den Zeilen in Wörter Arrays umgewandelt, danach die Arrays mit `explode` in weitere Zeilen erweitern und abschließend MapReduce mit `lit` und `groupBy` gemacht.

```
%pyspark
from pyspark.sql.functions import explode, split, lit, trim, regexp_replace

# read text file
df = spark.read.text('hdfs://141.100.62.85:9000/data/lorem/lorem.txt')

# remove end of line whitespace
df = df.withColumn("value",trim(df.value))
# filter rows with empty strings
df = df.filter("value != ''")
# remove special characters
df = df.withColumn("value", regexp_replace("value", '[^a-zäüößA-ZÄÜÖ ]+', ''))
#df = df.withColumn("value",split(df.value,'\s+|,\s+|\.\s+|\!\s')) # split by "," and "whitespace"
# split by whitespace
df = df.withColumn("value",split(df.value,' '))
# expand list into rows
df = df.select(explode("value").alias("key"))
# add new column with 1's
df = df.withColumn("value",lit(1))
# sum counts
#df = df.groupBy("key").sum("value").select(col("key"),col("sum(value)").alias("value"))
df = df.groupBy("key").sum("value")
# rename
df = df.withColumnRenamed('sum(value)', 'value')
# order by value
df = df.orderBy("value", ascending=False)

# show result
df.show()
```

```
+--------+-----+
|     key|value|
+--------+-----+
|      et|   57|
|   dolor|   32|
|     sed|   27|
|    diam|   27|
|     sit|   26|
|   ipsum|   26|
```

Aufgabe 2

Wir verbinden uns mittels ssh auf:

```
141.100.62.87
```

dort haben wir eine `tmux` session mittels

```
tmux -S /tmp/smux new -s amo
```

erstellt, auf die wir uns dann mittels

```
tmux -S /tmp/smux attach -t amo
```

gemeinsam verbinden und mit netcat arbeiten können.

unterschiede in den Parameters

Bei Output unterscheidet man zwischen folgenden Parameters:

- **append**: Nur neue Zeilen werden in den Output geschrieben. Exklusiv für de Verwendung ohne Aggregationen.
- **complete**: Alle Zeilen werden jedes mal in den Output geschrieben. Exklusiv für die Verwendung mit Aggregationen.
- **update**: Nur veränderte Zeilen werden in den Output geschrieben. Ohne Aggregation wie **append**.