Przetwarzanie i analiza dużych zbiorów danych

November 24, 2020

Analiza skupień w kontekście danych masowych

Barbara Morawska 234096
Andrzej Sasinowski 234118
Marcin Markiewicz 234090

Program został wykonany w języku Python przy użyciu Apache Spark. Program na podstawie danych wczytanych z pliku 4.txt wyznacza przy pomocy zaimplementowanego algorytmu A-priori reguły asocjacyjne dla dwójek oraz trójek produktów, a następnie wypisuje po 5 reguł z dwójek i trójek o najwyższym współczynniku ufności. W przypadku równych współczynników zastosowany jest porządek leksykograficzny według poprzedników.

Kod programu:

[3]:
```
import findspark
findspark.init()

from operator import add
import pyspark
import itertools
```

[4]:
```
def count_elements(elements, threshold_supp):
    elements_counted = elements.map(lambda key: (key, 1)).reduceByKey(add)
    elements_counted_filtered = elements_counted.filter(lambda element_dict:␣
 ↪element_dict[1] >= threshold_supp)
    return dict(elements_counted_filtered.collect())


def calculate_supp(session_data, elements, cardinality, threshold_supp):
    n_element_set = {}
    for combination in itertools.combinations(sorted(elements), cardinality):
        n_element_set[combination] = 0

    for session in session_data.toLocalIterator():
        session_filtered = sorted([element for element in session if element in␣
 ↪elements])
        for combination in itertools.combinations(session_filtered,␣
 ↪cardinality):
            if combination in n_element_set:
```

```
                n_element_set[combination] += 1

    n_element_set_filtered = filter(lambda element_dict: element_dict[1] >=␣
↪threshold_supp, n_element_set.items())
    return dict(n_element_set_filtered)
```

[5]:
```
def two_elements_case(element_set, element_subset):
    rules = []
    for key, value in element_set.items():
        ## X --> Y, Y --> X
        rules.append((list([key[0], key[1]]), value / element_subset[key[0]]))
        rules.append((list([key[1], key[0]]), value / element_subset[key[1]]))
    return rules


def three_elements_case(element_set, element_subset):
    rules = []
    for key, value in element_set.items():
        ## X,Y --> Z, X,Z --> Y, Y,Z --> Z
        for combination in itertools.combinations(key, 2):
            if combination in element_subset:
                p = list(combination)
                q = list(set(key) - set(combination))
                rules.append((p, q, value / element_subset[combination]))
    return rules


def assembly_rules(element_set, element_subset, set_cardinality):
    get_function = {2: two_elements_case, 3: three_elements_case}
    which_case = get_function.get(set_cardinality)
    rules = which_case(element_set, element_subset)
    rules.sort(reverse=True, key=lambda x: x[set_cardinality-1])
    return rules
```

[ ]:
```
DEFAULT_THRESHOLD_SUPP = 100

## Prepare data
sc = pyspark.SparkContext.getOrCreate()
session_data = sc.textFile('4.txt').map(lambda line: line[:-1].split(' '))
all_elements = session_data.flatMap(lambda element: element)

## 1-element supp
elements_single = count_elements(all_elements, DEFAULT_THRESHOLD_SUPP)

## 2-elements supp
elements_double = calculate_supp(session_data, elements_single, 2,␣
↪DEFAULT_THRESHOLD_SUPP)
```

```
elements_double_unique = set(itertools.chain.from_iterable(elements_double))

## 3-elements supp
elements_triple = calculate_supp(session_data, elements_double_unique, 3,␣
↪DEFAULT_THRESHOLD_SUPP)

## Assembly rules and calculate confidence
rules_double = assembly_rules(elements_double, elements_single, 2)
rules_triple = assembly_rules(elements_triple, elements_double, 3)

# Print number of of 1-element sets
print("Liczba zbiorów jednoelementowych: {}".format(len(elements_single)))

# Get five best association rules for both sets

print("\n5 reguł asocjacyjnych dla dwójek o najwyższym współczynniku ufności: ")
for rule in rules_double[:5]:
    print(rule)

print("\n5 reguł asocjacyjnych dla trójek o najwyższym współczynniku ufności: ")
for rule in rules_triple[:5]:
    print(rule)

sc.stop()
```

Wyniki:

```
Liczba zbiorów jednoelementowych: 647

5 reguł asocjacyjnych dla dwójek o najwyższym współczynniku ufności:
(['DAI93865', 'FRO40251'], 1.0)
(['GRO85051', 'FRO40251'], 0.999176276771005)
(['GRO38636', 'FRO40251'], 0.9906542056074766)
(['ELE12951', 'FRO40251'], 0.9905660377358491)
(['DAI88079', 'FRO40251'], 0.9867256637168141)

5 reguł asocjacyjnych dla trójek o najwyższym współczynniku ufności:
(['DAI23334', 'ELE92920'], ['DAI62779'], 1.0)
(['DAI31081', 'GRO85051'], ['FRO40251'], 1.0)
(['DAI55911', 'GRO85051'], ['FRO40251'], 1.0)
(['DAI62779', 'DAI88079'], ['FRO40251'], 1.0)
(['DAI75645', 'GRO85051'], ['FRO40251'], 1.0)
```