Principles of Computer Science II

Large Scale Computation with Apache Spark

Ioannis Chatzigiannakis

Sapienza University of Rome

Lecture 18

Profile most-frequent k-mer

CGGGGCTGGGTCGTCACATTCCCCTTTCGATA
TTTGAGGGTGCCCAATAACCAAAGCGGACAAA
GGGATGCCGTTTGACGACCTAAATCAACGGCC
AAGGCCAGGAGCGCCTTTGCTGGTTCTACCTG
AATTTTCTAAAAAGATTATAATGTCGGTCCTC
CTGCTGTACAACTGAGATCATGCTGCTTCAAC
TACATGATCTTTTGTGGATGAGGGAATGATGC

- Seven (7) 32-nucleotide DNA sequences
- A "secret" pattern P=ATGCAACT of length $l = 8$ implanted.
- Can you reconstruct P by analyzing the DNA sequences?

An example

CGGGGCTATGCAACTGGGTCGTCACATTCCCCTTTCGATA
TTTGAGGGTGCCCAATAAATGCAACTCCAAAGCGGACAAA
GGATGCAACTGATGCCGTTTGACGACCTAAATCAACGGCC
AAGGATGCAACTCCAGGAGCGCCTTTGCTGGTTCTACCTG
AATTTTCTAAAAAGATTATAATGTCGGTCCATGCAACTTC
CTGCTGTACAACTGAGATCATGCTGCATGCAACTTTCAAC
TACATGATCTTTTGATGCAACTTGGATGAGGGAATGATGC

- The same DNA sequences with the implanted pattern ATGCAACT
- Can you spot the locations of the implanted pattern?

An example

CGGGGCTATGCAACTGGGTCGTCACATTCCCCTTTCGATA
TTTGAGGGTGCCCAATAAATGCAACTCCAAAGCGGACAAA
GGATGCAACTGATGCCGTTTGACGACCTAAATCAACGGCC
AAGGATGCAACTCCAGGAGCGCCTTTGCTGGTTCTACCTG
AATTTTCTAAAAAGATTATAATGTCGGTCCATGCAACTTC
CTGCTGTACAACTGAGATCATGCTGCATGCAACTTTCAAC
TACATGATCTTTTGATGCAACTTGGATGAGGGAATGATGC

- Same as before but showing the implant locations.
- Devise an MapReduce algorithm to automatically identify the implanted pattern
- Length l is known.
- Sequence: `https://goo.gl/xN7WvE`

Profile most-frequent k-mer

```
from pyspark import SparkContext, SparkConf
from operator import add

conf = SparkConf().setAppName("Profile").setMaster("local")
sc = SparkContext(conf=conf)

val raw_data = sc.wholeTextFiles("pattern.txt")

def splitLine(line):
  pairs = []
  if len(line) > 1:
    for symbol in range(0, len(line)-8):
      pairs.append((line[symbol:symbol+8], 1))

  return pairs

pairs = raw_data.flatMap(splitLine) \
                .reduceByKey(add) \
                .sortBy(lambda a: -a[1])
print(pairs.take(5))
```

Profile most-Frequent first appearing k-mer

CGGGGCTGGGTCGTCACATTCCCCTTTCGATA
TTTGAGGGTGCCCAATAACCAAAGCGGACAAA
GGGATGCCGTTTGACGACCTAAATCAACGGCC
AAGGCCAGGAGCGCCTTTGCTGGTTCTACCTG
AATTTTCTAAAAAGATTATAATGTCGGTCCTC
CTGCTGTACAACTGAGATCATGCTGCTTCAAC
TACATGATCTTTTGTGGATGAGGGAATGATGC

- Identify most-frequent first-appearing k-mer
- In first line, 3-mer CGG appears before GGC, TGG, GTC, ...

Profile most-frequent k-mer

```
from pyspark import SparkContext, SparkConf
from operator import add

conf = SparkConf().setAppName("Profile").setMaster("local")
sc = SparkContext(conf=conf)

raw_data = sc.textFile("yeast_chr1.txt")

def splitLine(line):
  pairs = []
  for symbol in range(0, len(line)-6, 3):
      for second in range(symbol+3, len(line)-6, 3):
          pairs.append(((line[symbol:symbol+3],
                         line[second:second+3]), 1))

  return pairs

pairs = raw_data.flatMap(splitLine) \
                .reduceByKey(add) \
                .sortBy(lambda a: -a[1])
print(pairs.take(10))
```