COMP9313: Big Data Management

Sample Exam Solutions

Question 1 HDFS

- Explain the difference between NameNode and DataNode.
  - Answer: see next page
- Given a file of 500MB, let block size be 150MB, and replication factor=3. How much space do we need to store this file in HDFS? Why?
  - Answer: 500MB * 3 = 1500MB, as for each block we need to store 3 replicas. There are 4 blocks for this file, 3 with 150MB each and 1 with 50MB.

	NameNode	DataNode
Quantity	One	Multiple
Role	Master	Slave
Stores	Metadata of files	Blocks
Hardware Requirements	High Memory	High Volume Hard Drive
Failure rate	Lower	Higher
Solution to Failure	Secondary NameNode	Replications

Question 2 Spark

- Given a large text file, your task is to find out the top-k most frequent co-occurring term pairs. The co-occurrence of (w, u) is defined as: u and w appear in the same line (this also means that (w, u) and (u, w) are treated equally). Your Spark program should generate a list of ***k*** key-value pairs ranked in descending order according to the frequencies, where the keys are the pair of terms and the values are the co-occurring frequencies (**Hint:** you need to define a function which takes an array of terms as input and generate all possible pairs).

```
textFile = sc.textFile(inputFile)
words = textFile.map(lambda x: x.lower().split())

// fill your code here, and store the result in a pair RDD avgLen

avgLen.take(k)
```

Answer

```
textFile = sc.textFile(inputFile)
words = textFile.map(lambda x: x.lower().split())

def gen_pair(x):
    pairs = []
    for i in range(len(x)):
        for j in range(i+1, len(x)):
            if x[i] < x[j]:
                pairs.append(((x[i],x[j]), 1))
            else:
                pairs.append(((x[j],x[i]), 1))
    return pairs

pairs = word.flatMap(lambda x: gen_pair(x)).reduceByKey(lambda x, y: x + y)
topk = pairs.map(lambda x: (x[1],x[0])).sortByKey(False).map(lambda x: (x[1], x[0]))

avgLen.take(k)
```

Question 3 Finding Similar Items

Suppose we wish to find similar sets, and we apply locality-sensitive hashing with k=5 and l=2.

If two sets had Jaccard similarity 0.6, what is the probability that they will be identified in the locality-sensitive hashing as candidates (i.e. they hash at least once to the same super-hash)? You may assume that there are no coincidences, where two unequal values hash to the same hash value.

Answer: $1-(1-0.6^5)^2 = 0.149$

Question 4 Mining Data Streams

Suppose we are maintaining a count of 1s using the DGIM method. We represent a bucket by (i, t), where i is the number of 1s in the bucket and t is the bucket timestamp (time of the most recent 1).

Consider that the current time is 200, window size is 60, and the current list of buckets is: (16, 148) (8, 162) (8, 177) (4, 183) (2, 192) (1, 197) (1, 200). At the next ten clocks, 201 through 210, the stream has 0101010101. What will the sequence of buckets be at the end of these ten inputs?

Answer

There are 5 1s in the stream. Each one will update to windows to be: [each step 2 marks]

- (1) (16, 148)(8, 162)(8, 177)(4, 183)(2, 192)(1, 197)(1, 200), (1, 202)

=> (16, 148)(8, 162)(8, 177)(4, 183)(2, 192)(2, 200), (1, 202)

- (2) (16, 148)(8, 162)(8, 177)(4, 183)(2, 192)(2, 200), (1, 202), (1, 204)
- (3) (16, 148)(8, 162)(8, 177)(4, 183)(2, 192)(2, 200), (1, 202), (1, 204), (1; 206)

=> (16, 148)(8, 162)(8, 177)(4, 183)(2, 192)(2, 200), (2, 204), (1, 206)

=> (16, 148)(8, 162)(8, 177)(4, 183)(4, 200), (2, 204), (1, 206)

- (4) Windows Size is 60, so (16,148) should be dropped.

(16, 148)(8, 162)(8, 177)(4, 183)(4, 200), (2, 204), (1, 206), (1, 208) => (8, 162)(8, 177)(4, 183)(4, 200), (2, 204), (1, 206), (1, 208)

- (5) (8, 162)(8, 177)(4, 183)(4, 200), (2, 204), (1, 206), (1, 208), (1, 210)

=> (8, 162)(8, 177)(4, 183)(4, 200), (2, 204), (2, 208), (1, 210)

Question 5 Recommender Systems

Consider three users u_1, u_2, and u_3, and four movies m_1, m_2, m_3, and m_4. The users rated the movies using a 4-point scale: -1: bad, 1: fair, 2: good, and 3: great. A rating of 0 means that the user did not rate the movie. The three users' ratings for the four movies are: $u_1 = (3, 0, 0, -1)$, $u_2 = (2, -1, 0, 3)$, $u_3 = (3, 0, 3, 1)$

- Which user has more similar taste to u_1 based on cosine similarity, u_2 or u_3? Show detailed calculation process.
  - Answer: sim(u1, u2) = (3*2 -1*3)/(sqrt(10)*sqrt(14)) ≈ 0.2535, sim(u1, u3) = (3*3-1*1)/(sqrt(10)*sqrt(19)) ≈ 0.5804. Thus u3 is more similar to u1.
- User u_1 has not yet watched movies m_2 and m_3. Which movie(s) are you going to recommend to user u_1, based on the user-based collaborative filtering approach? Justify your answer.
  - Answer: You can use either cosine similarity or Pearson correlation coefficient to compute the similarities between users. However, the conclusion should be that only m3 is recommended to u1.