Module 1 Quiz

Quiz, 10 questions

1 point

1.

Assume that MapReduce(f, g, data), performs map function f on the input data, followed by reduce function g, as introduced in Lecture 1.1. What does the function *mystery*, provided below in pseudocode notation, compute?

Hint: Determining the omitted type information might be helpful.

```
1  f(k, v) {
2    return [(v, (k, 1))]      // a collection of one pair
3  }
4
5  g(pair1, pair2) {
6    return (pair1[0] + pair2[0], pair1[1] + pair2[1])
7  }
8
9  mystery(data) {
10   for (k, (v1, v2)) in MapReduce(f, g, data)
11     add (k, v1 / v2) to results
12   return results
13 }
```

- (●) A. For each input value, the average corresponding input key.
- ( ) B. For each input key, the average corresponding input value.
- ( ) C. For each input value, the average position in the ordering of input keys.
- ( ) D. For each position in the ordering of input keys, the average of the previous input values.

---

1 point

Module 1 Quiz

Quiz, 10 questions

2.

Consider the following pseudo-code using MapReduce. What does *mystery2* print when its input is a collection with a single key/value pair entry?

Hint: Determining the omitted type information might be helpful.

```
1  m(k, v) {
2    vs = v.tokenize("\n")
3    res = []
4    for i in 0 ... vs.length - 1
5      ws = vs[i].tokenize(" ")
6      for w in ws
7        res += ([w, k], [i + 1])
8    return res
9  }
10
11 r(v1, v2) {
12   res = []
13   for v in v1
14     res += v
15   for v in v2
16     res += v
17   return res
18 }
19
20 mystery2(data) {
21   for (k, v) in MapReduce(m, r, data)
22     print(k[0] + " = " + v)
23 }
24
```

- A. For each input value, all the words that occur in each line.
- B. For each line in the input value, the average length of the words it contains.
- C. For each line in the input value, the number of words it contains.
- D. For each input value, the words and the line numbers they appear in.
- E. For each input value, the words and the line numbers they appear in in sorted order.

---

1 point

3.

Which of the following statements are true?

Module 1 Quiz

Quiz, 10 questions

- [x] A. Hadoop is able to provide strong fault tolerance because map-reduce is a functional operation.
- [ ] B. Hadoop is used to speed up tasks executing on a single computer.
- [ ] C. The user of Hadoop is responsible for specifying how the computers in the network communicate with each other.

---

1 point

4.

Suppose we want to use Hadoop to perform a word count on the following words, given as a sequence of key-value pairs: (the, 1), (dog, 1), (chased, 1), (the, 1), and (cat, 1). Which of the following could be the output of the "group" step of the computation? Assume no reduction operation can occur during the "group" step.

- [ ] A. Group 1: (the, 1). Group 2: (dog, 1). Group 3: (chased, 1). Group 4: (the, 1). Group 5: (cat, 1).
- [ ] B. Group 1: (the, 2). Group 2: (dog, 1). Group 3: (chased, 1). Group 4: (cat, 1).
- [x] C. Group 1: (the, 1) and (the, 1). Group 2: (dog, 1). Group 3: (chased, 1). Group 4: (cat, 1).

---

1 point

5.

Which of the following statements are true?

- [ ] A. Hadoop is a generalization of Spark.
- [x] B. Spark benefits from using nodes with large memory.
- [ ] C. Spark only supports eager or strict evaluation. (It does not use lazy evaluation.)

Module 1 Quiz

Quiz, 10 questions

1 point

6.

Which of the following are terminal operations in Spark?

- [ ] A. Map
- [x] B. Reduce
- [ ] C. Filter
- [x] D. Collect

---

1 point

7.

Why might the notion of an “inverse document frequency” be important in determining the similarity between two arbitrary documents in a corpus?

- (•) A. The inverse document frequency ensures that words appearing across many documents are discounted in comparison to words that are unique to a few documents.
- ( ) B. The inverse document frequency adds prominence to words that are common to many documents, enabling a programmer to compute similarity metrics more precisely.
- ( ) C. The inverse document frequency allows the programmer to more easily apply the MapReduce model of computation to a large corpus of documents.

---

1 point

8.

For this question, you are encouraged to search the Internet and read about topics relevant to elementary text-mining and tf-idf.

Module 1 Quiz

Quiz, 10 questions

Which of the following might serve as a valid means of computing which document is most similar to document D_1 in a corpus of text documents? For simplicity, assume all relevant tf-idf weights have been appropriately normalized and computed, and that resources (time, memory, secondary storage, etc) are not issues of concern.

I. For each document in the corpus, use a vector representation of the appropriate tf-idf weights to compute its cosine similarity to D_1. Those documents yielding the highest cosine similarity to D_1 are considered “most similar”.

II. For each document in the corpus, find the sum of its tf-idf weights corresponding to the words that appear in D_1. Those documents yielding the highest sum of relevant tf-idf weights are considered “most similar”.

III. For each document in the corpus, compare its total word count to the word count of D_1. Those documents yielding the smallest absolute difference in word counts are considered “most similar”.

- (●) A. I and II
- ( ) B. II and III
- ( ) C. III only
- ( ) D. I and III

---

1 point

9.

In Spark, what does the reduceByKey transformation do?

- (●) A. For a set of (key, value) pairs, groups all values that have the same key and then applies a reduction operator to collapse those values into a single value. A single (key, value) pair is then emitted per key.
- ( ) B. For a set of (key, value) pairs, groups all values that have the same key. A single pair of a key and the list of all values is then emitted.
- ( )

Module 1 Quiz

Quiz, 10 questions

C. For a set of (key, value) pairs, counts the number of (key, value) pairs for each unique key. For each unique key, emits a (key, value) pair that is that key with the number of pairs that had it.

D. For a set of (key, value) pairs and a given key, removes all (key, value) pairs that have that same key.

---

1 point

10.

Why is Page Rank an algorithm that fits well with Spark?

- Spark offers Page Rank transformations specifically for supporting the Page Rank algorithm (e.g. join).
- Spark is optimized for iterative, in-memory workloads. Page Rank is an example of one.
- Page Rank was originally designed to run on top of Spark, and so algorithmically fits well with the provided transformations.
- Very few distributed programming models could support a distributed join operation.

I, **Fan Yang**, understand that submitting work that isn't my own may result in permanent failure of this course or deactivation of my Coursera account.

Learn more about Coursera's Honor Code

Submit Quiz