Show Submission Credentials

OPE - Spark programming Spark programming OPE

Introduction

Introduction

Information

Learning Objectives

- Utilize the suitable map and reduce operations in Spark to transform and aggregate data.
- Utilize suitable functional operations in Scala to transform data structures.
- Identify the use case of a JOIN operation and utilize JOIN to combine and transform different RDDs in Spark.
- Identify the use case of the cache() method and utilize it to improve the performance of a Spark application.

Welcome to the Spark OPE. In this exercise, you will get hands-on experience with writing Apache Spark code in Scala using TDD. Your task will be to create an inverted index over a set of documents using the Spark programming model. The purpose of an inverted index is similar to the glossary at the end of books. They are used to quickly find all the documents that contain a given word. Figure 1 shows an example of an inverted index.

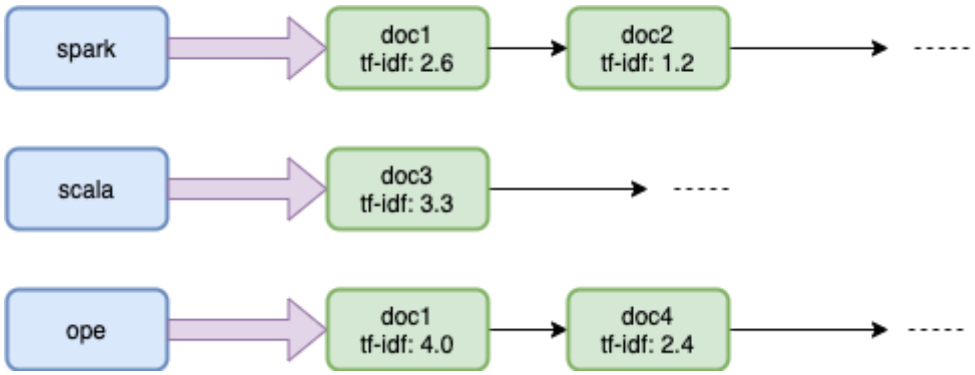


Figure 1: An example of an inverted index

You will be provided with the Scalatest test cases. This experience will benefit you in the upcoming Spark project.

Please remember to fill out the pre-quiz (**credit received on completion**) below using the token that was provided in your email prior to starting the OPE Task. This will unlock the instructions for the actual OPE task so that you can continue. Use the token provided by the submitter on Cloud9 to unlock the post-quiz below. Be sure to fill out the graded post-quiz (**graded on correctness**) after you are done with the OPE task on Cloud9.

Information

You will receive your token for the pre-quiz about 20 minutes prior to the OPE task. Please remember when switching roles to set the account as our course AWS ID: 752574329361, NOT your account.

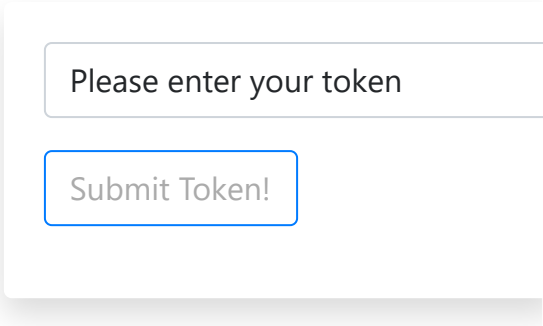
If your token is not unlocking your Pre-Quiz, please post privately to Piazza.

Danger

Make sure *all* team members submit the project so that the score is reflected in Sail(). Do not leave Cloud9 until all team members see their score.

Also, only one student should submit at a time.

Pre-OPE Quiz



Glossary

We summarize the terminologies used in this writeup and what they mean in this OPE. It is helpful to go through the terminologies if this is the first time you hear of the inverted index.

Terminology	Short Description	Description
token	normalized word, term	in this context, it is a word split by spaces and is lowercase.
document	document	think of it as an individual tweet, or an individual Wikipedia article.

corpus	a collection of documents.	a collection of documents.
TF	term frequency	the number of times a token occurs in one document.
DF	document frequency	the number of documents a token occurs in.
IDF	inverse document frequency	a measure of whether a term is common or rare in a given document corpus
inverted index	inverted index	postings lists indexed by terms

Task 1

Information

Once you join the OPE session on Cloud9, *one* of the members from your OPE team should run the below command to setup the OPE environment in the Cloud9 Terminal:

```
wget https://s3.amazonaws.com/cmucc-instance-launcher/env_setup_oneshot_spa
rk_ope.sh && sh env_setup_oneshot_spark_ope.sh
```

Make sure you go through the README.md file carefully before working on your tasks. Auto saving of files is not enabled by default, so make sure to save before running a test else you might run old code.

:

As described in the introduction, in the OPE you'll have to create an inverted index of terms in a corpus, with TF-IDF as the scoring metric.

TF-IDF is short for term frequency–inverse document frequency. Term frequency is how many times a word appears in a given document.

However, term frequency by itself is not a good information retrieval metric because popular words (such as `the`, or `and`) will have a higher score than other more relevant words in a document simply because they appear in all documents. Therefore, the inverse document frequency tries to penalize frequent words in the corpus.

In short, TF-IDF reflects how important a word is to a document in a corpus.

$$tf\_idf(t,d,C) = \log\big(1 + f(t,d)\big) * \left[\log\left(\frac{n}{1 + df(t,C)}\right) + 1\right]$$

$$t: token$$
$$d: document$$
$$C: corpus$$
$$f: frequency$$
$$df: document\ frequency$$

Figure 2: TF-IDF formula

Note that there are slightly different variations of TF-IDF. That being said, you don't need to worry about the formula as it will be provided in the starter code.

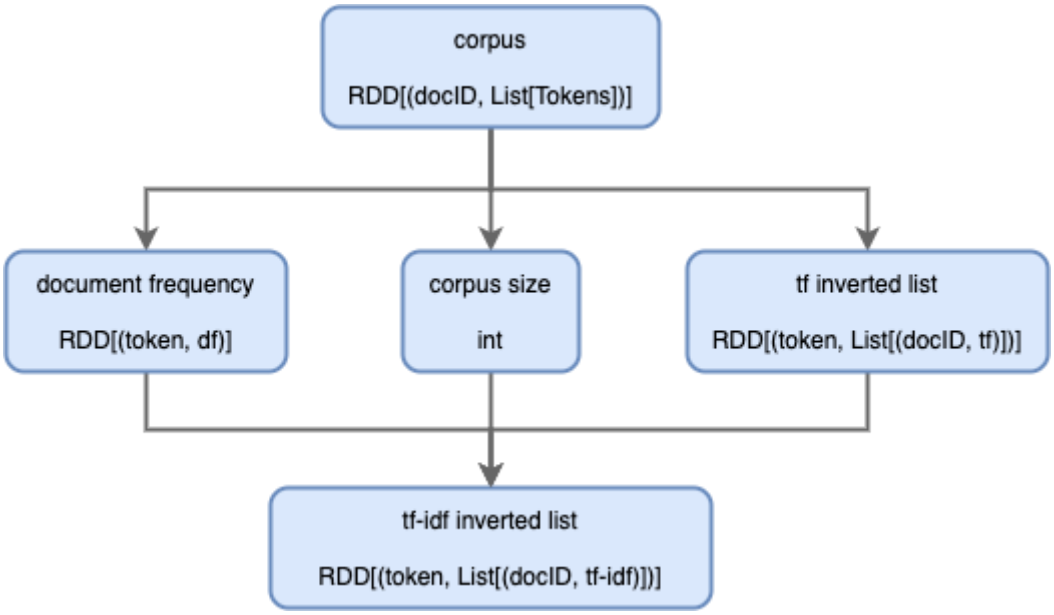


Figure 3: TF-IDF design

In the first task, you will compute the document frequency. Task 2 will have you compute the term frequency inverted list. Task 3 will have you join all three to compute the TF-IDF inverted index. Finally, Task 4 will ask a more conceptual question.

As a side note, there are more efficient ways of computing the inverted index with TF-IDF. This approach is for educational purposes.

Information

Don’t worry, most of the heavy lifting is already done for you in the provided template. You just need to complete the code where you see the placeholder " `???` ".

Task 1

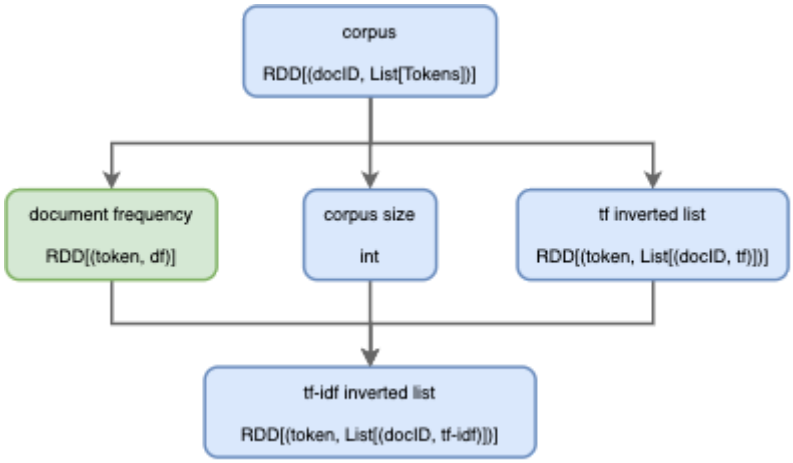


Figure 4: Task 1

For this task you will compute the document frequency RDD. For that you need to complete function `Task1#documentFrequency`. You don't have to worry how the RDD is loaded, and tokenized. You can assume it is in the right format.

Figure 5: Task 1 example

Information

- Remember that document frequency is in how many documents a token is present in.
  - Note that if a document has the tokens:

    ```
    stay, hungry, stay, foolish
    ```

    `stay` should only be counted once for that document.
  - Read the README file before starting to code.

Danger

The sections of the code that you should implement are marked with a `???`.

DO NOT modify any other sections of `src/main/scala/Task1.scala` or any other files.

Information

Once you are confident of your solution:

1. Save the file (autosave is not enabled by default)
2. run `./test-checker task1`

Danger

If you did not get to complete Task 1, and the OPE is over, before closing Cloud9, make sure *all* students submitted the code and can see their score in the submission table.

```
`./submitter-ope`
```

Concurrent submissions might produce unexpected grading results - one student should submit at a time.

Task 2

Task 2

In Task 1 you built an RDD of document frequencies. Task 2 has you build an inverted index of term frequencies.

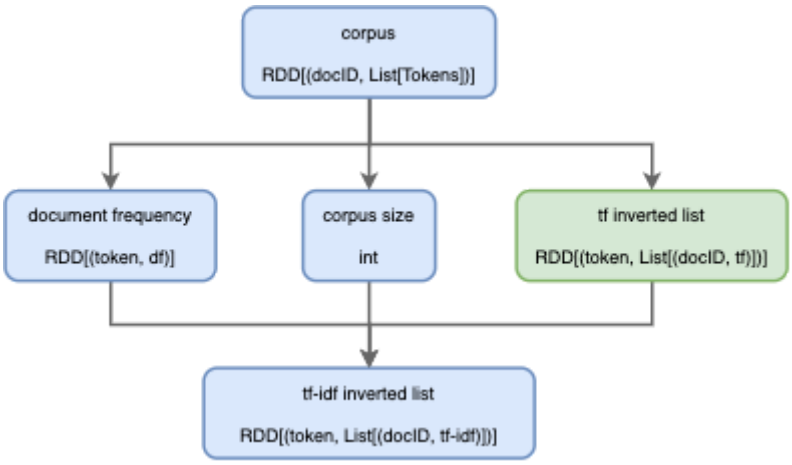


Figure 6: Task 2

In this task, however, we provide you with the necessary Spark code:

```
  /**
    * Given an RDD (docID, List[Tokens]) it returns an RDD of [(Token, Iterable
[DocID, Frequency])]
    *
    * The frequency is the number of occurrences of a token in the given documen
t.
    *
    * For example, given the following RDD:
    *
    *           "file_1", ["read", "fun", "fun", "fun"]
    *           "file_2", ["read", "fun", "primer", "primer"]
    *           "file_3", ["read", "idea"]
    *
    * The output RDD should be:
    *
    *           "read",   [(file_1, 1), (file_2, 1), (file_3, 1)]
    *           "fun",    [(file_1, 3), (file_2, 1)]
    *           "primer", [(file_2, 2)]
    *           "idea",   [(file_3, 1)]
    *
    * @param tokenizedCorpus the RDD of (docID, List[Tokens]).
    * @return an inverted index of term frequencies.
    */
  def tokenFrequencyInvertedList(
      tokenizedCorpus: RDD[(String, Seq[String])]): RDD[(String, Iterable[(Strin
g, Int)])] = {
    tokenizedCorpus.flatMap {
        case (documentId, tokens) => docTokens2tokenDocFreq(documentId, tokens)
      }
      .groupByKey()
  }
```

Your task is to complete the function `docTokens2tokenDocFreq`. It takes a document id and a list of tokens, and it returns `(token, (docID, frequency))`.

Figure 7: Task 2 example

Information

We have provided you with a helper function that given a list of tokens, it returns a list of `List[(token, frequency)]`.

Note that all elements in the output list will have the same document id.

Information

Once you are confident of your solution:

1. Save the file (autosave is not enabled by default)
2. run `./test-checker task2`

Danger

If you did not get to complete Task 2, and the OPE is over, before closing Cloud9, make sure *all* students submitted the code and can see their score in the submission table.

```
`./submitter-ope`
```

Concurrent submissions might produce unexpected grading results - one student should submit at a time.

Task 3

Task 3

In Task 1 you built an RDD of document frequencies, in Task 2 you built an inverted index of term frequencies. Finally, in Task 3 you'll join both RDDs, and compute the TF-IDF score.

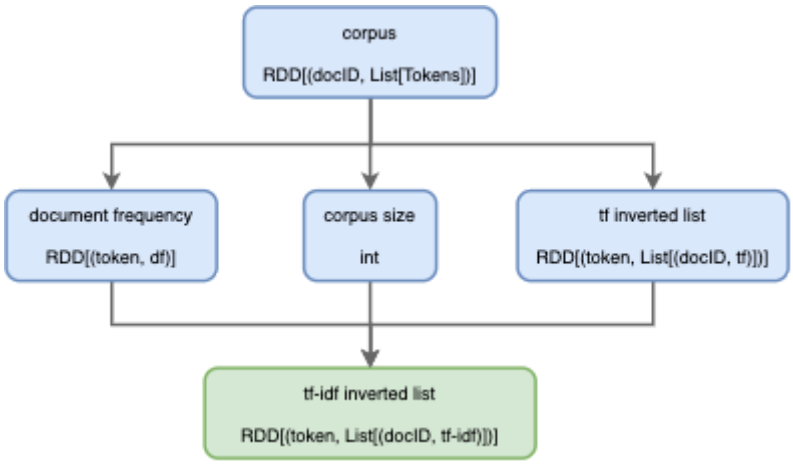


Figure 8: Task 3

Your task is to complete the function `invertedIndex` that takes the document frequency RDD from Task 1, the inverted index with term frequencies from Task 2, the total number of documents in the corpus, and outputs the inverted index of TF-IDF.

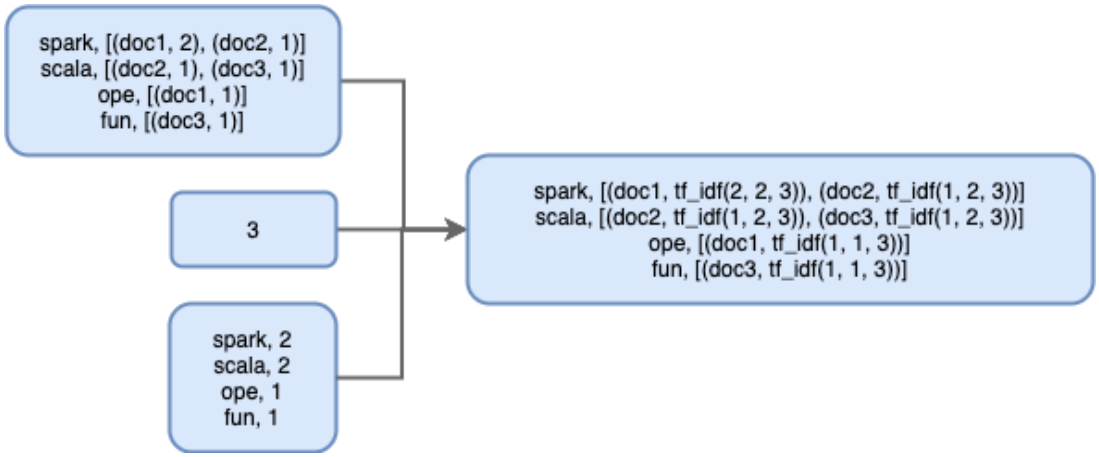


Figure 9: Task 3 example

Information

We have provided you with a helper function that given a term frequency, document frequency, and total documents in the corpus, it returns the TF-IDF score. You *don't* need to modify that function.

Information

Once you are confident of your solution:

1. Save the file (autosave is not enabled by default)
2. run `./test-checker task3`

Danger

If you did not get to complete Task 3, and the OPE is over, before closing Cloud9, make sure *all* students submitted the code and can see their score in submission table.

```
`./submitter-ope`
```

Concurrent submissions might produce unexpected grading results - one student should submit at a time.

Task 4

Task 4

Almost there! Don't worry, all of the heavy lifting is already done. Task 4 pieces all of the tasks together.

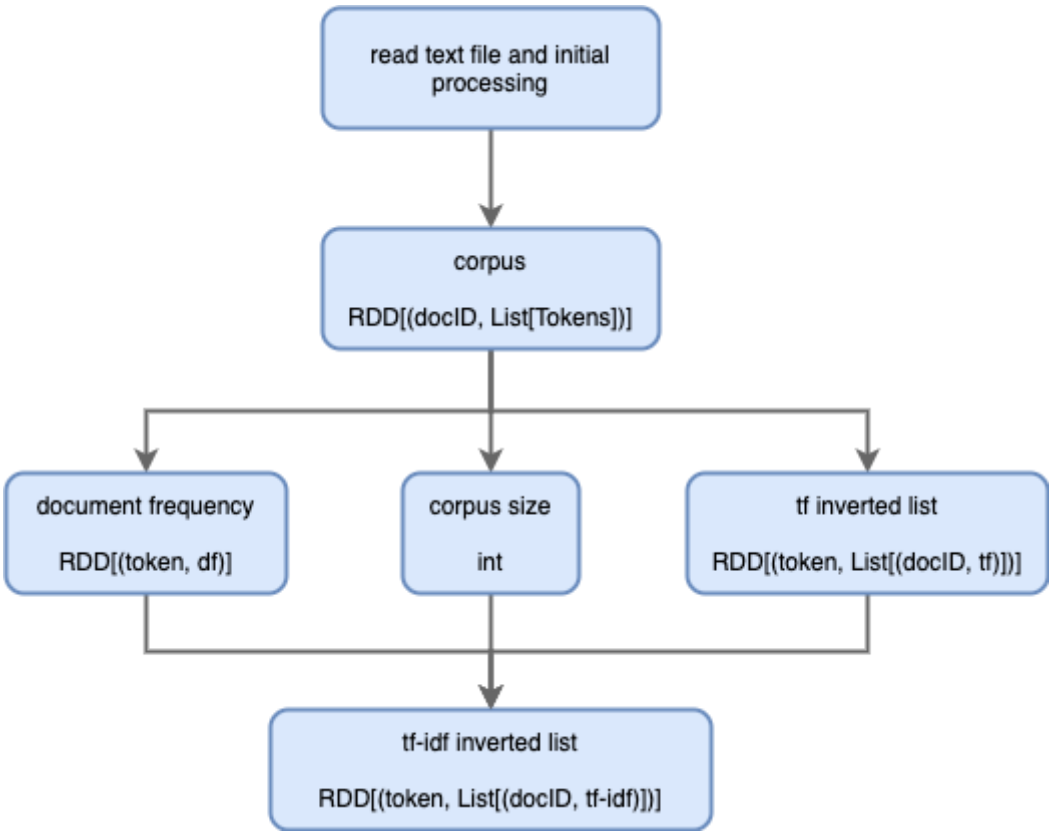


Figure 10: Task 4

Notice that we added a new RDD. This is the raw files that are read, and tokenized (split into tokens, is lowercase, and stop-words removed).

Luckily for you, we have provided you with all the code for that. Your task is to think about what RDD would improve the performance of the job if the RDD is cached, and cache it.

```
def buildInvertedIndex(
    pathToTextCorpus: String,
    sc: SparkContext): RDD[(String, Iterable[(String, Double)])] = {
  val corpus = readTextFile(pathToTextCorpus, sc)
  val tokenizedCorpus = tokenize(corpus)

  val n = numberOfDocuments(tokenizedCorpus)
  val df = Task1.documentFrequency(tokenizedCorpus)
  val tf = Task2.tokenFrequencyInvertedList(tokenizedCorpus)

  Task3.invertedIndex(tf, df, n)
}
```

Warning

You only need to add `.cache()` to one of the existing RDD. For example, if you had the following RDD that you wanted to cache:

```
val rdd = sc.textFile(...)
```

You only need to add .cache() at the end like this:

```
val rdd = sc.textFile(...).cache()
```

Information

Even though we provide you with the code, you are welcome to read it and to try to understand it. That being said, please don't modify anything other than adding `.cache()`.

Information

Once you are confident of your solution:

1. Save the file (autosave is not enabled by default)
2. run `./test-checker task4`

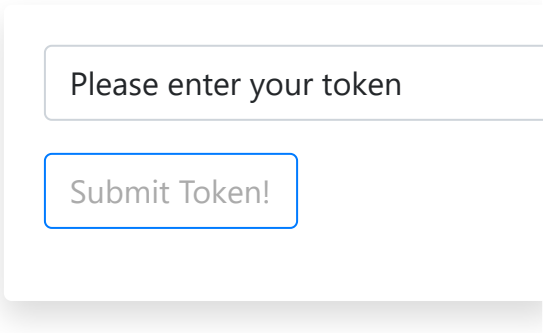
Danger

If you completed Task 4, or if you ran out of time, before closing Cloud9, make sure *all* students submitted the code and can see their score in submission table.

```
`./submitter-ope`
```

Concurrent submissions might produce unexpected grading results - one student should submit at a time.

Post-OPE Quiz


Please enter your token
Submit Token!