

HW4

Deadline: Apr. 23rd 5:59 P.M. (before class)

There are 4 options in this homework and you can pick one of them.

Option 1: Mahout KMeans

The K-Means algorithm is to cluster data points into different partitions based on some distance measures. In this part, you need to do K-Means algorithm in all the three platforms covered in this semester.

Data

Reuters-21578 is a collection of documents that appeared on Reuters newswire in 1987. The collection was released by Reuters and CGI in 1990 and now is available in the link below.

<https://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html>

Download and unzip reuters21578.tgz from the link above. Besides some descriptive files, you will find the collection contains 22 data files (reut2-XXX.sgm). **In this homework, you will need to use these 22 files for the further tasks.**

TODO

Part I. TF-IDF Representation

Create TF-IDF vector representing **sgm** documents using command line. A very detailed tutorial is given in the link below:

<https://mahout.apache.org/users/basics/creating-vectors-from-text.html>

Specifically, you will need two steps:

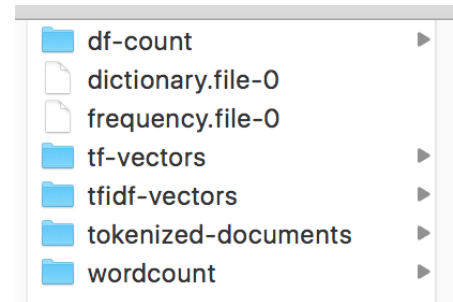
- Convert documents to SequenceFile format.
With the **mahout seqdirectory** command, you can convert text file to sequence file. The reason we must do this step is that Mahout command line takes sequence file (Writable) as input for most of its execution. Command example as follow,

```
$mahout seqdirectory -i <input> -o <output>
```

- Create a unigram TF-IDF vectors from the sequence file.
With the sequence file generated from step (1), we now can create TF-IDF vectors with **mahout seq2sparse** command. For example,

```
$mahout seq2sparse -i <input> -o <output> -ng 1
```

The output of the command contains several folders, including intermediate results such as tf-vectors and dictionary files such as dictionary.file-0. The tfidf-vectors folder should be the one you need for the further steps.



You may need to use `$mahout seqdumper` command if you want to understand what you have in the files. The available arguments are similar to the previous commands. Please check the appendix for more detail on description of each function.

Part II. K-Means Clustering

Use `$mahout kmeans` command line to find cluster centroids.

- Set **k=4**, **maxIter=10**, and distance function as
Check mahout kmeans command line:
<https://mahout.apache.org/users/clustering/k-means-commandline.html>
- **input:** The TF-IDF file generated from part I.
- **output:** The command line will generate a sequence file of output.
- When using this command, a path to a directory that stores clusters (-c) may also need to be specified.
- Use Euclidean distance as your distance measurement.

Part III. Report Results

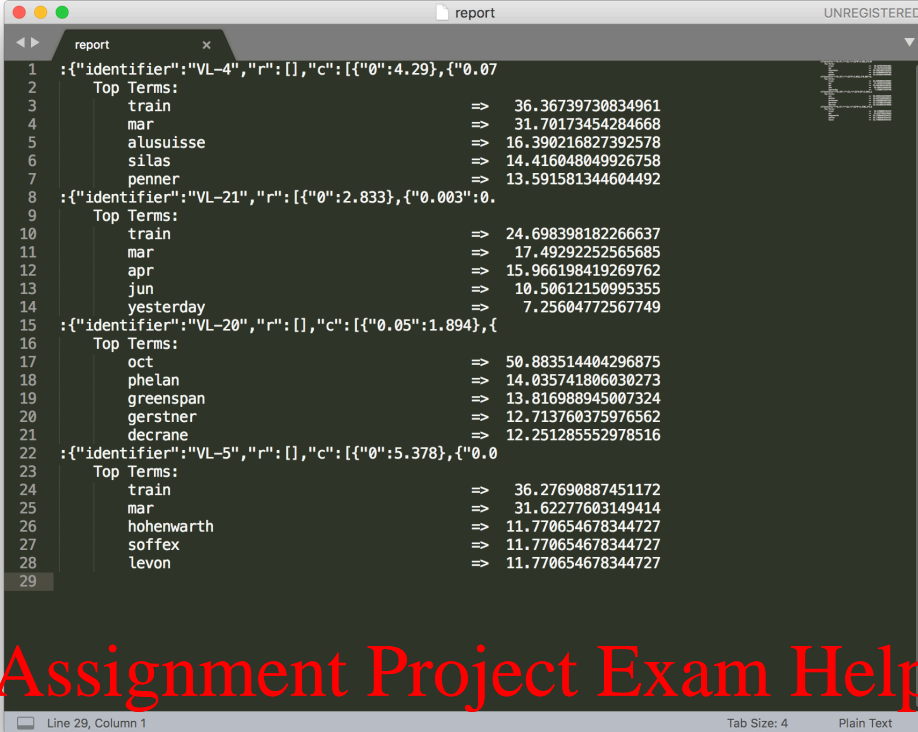
Use `$mahout clusterdump` command to report top 10 words in each cluster (Read appendix for the help menu).

Besides the usual arguments such as input file and output, please also include the following arguments for generating report:

- -d and -dt for specifying dictionary file and dictionary file type. If these 2 arguments are not specified, the result will only contain the index of words rather than words themselves.
- -b for specifying length of each centroid (e.g. -b 50). Without this argument, you will see a loooooong vector representing each centroid.

The output is similarly as the screenshot below.

¹ <https://mahout.apache.org/users/clustering/clusteringyourdata.html>



```
1 :{"identifier":"VL-4","r":[],"c":{"0":4.29},"0.07
2   Top Terms:
3     train      => 36.36739730834961
4     mar        => 31.70173454284668
5     alusuisse  => 16.390216827392578
6     silas      => 14.416048049926758
7     penner     => 13.591581344604492
8 :{"identifier":"VL-21","r":{"0":2.833},"0.003":0.
9   Top Terms:
10    train      => 24.698398182266637
11    mar        => 17.49292252565685
12    apr        => 15.966198419269762
13    jun        => 10.50612150995355
14    yesterday  => 7.25604772567749
15 :{"identifier":"VL-20","r":[],"c":{"0.05":1.894},{
16   Top Terms:
17    oct        => 50.883514404296875
18    phelan     => 14.035741806030273
19    greenspan  => 13.816988945007324
20    gerstner   => 12.713760375976562
21    decrane    => 12.251285552978516
22 :{"identifier":"VL-5","r":[],"c":{"0":5.378},"0.0
23   Top Terms:
24    train      => 36.27690887451172
25    mar        => 31.62277603149414
26    hohenwarth => 11.770654678344727
27    soffex     => 11.770654678344727
28    levon      => 11.770654678344727
29
```

Assignment Project Exam Help

<https://powcoder.com>

Submission

1. Mahout script for generating TF-IDF vectors for part I.
2. Mahout script for part II & III.
3. The result required in part III.

Add WeChat powcoder

Option 2: Hadoop KMeans

Use the same dataset in **Option 1**.

TODO

Part I. TF-IDF Representation

Same requirement with Part I in Option 1.

Part II. K-Means Clustering

Write a Hadoop MapReduce program to find centroids and its members with K-Means clustering. Your program must have the following **arguments**:

- **input**: String. Path to your input file/folder.
You can use the TF-IDF data converted from Mahout as your input file. However, you cannot use it directly. This is because this is a SequenceFile. Therefore, you need to convert the SequenceFile to text file so that your program can read it. The corresponding mahout command is `$mahout vectordump` (Read appendix for the help menu).
- **output**: String. Path to your output folder containing part-r-XXXXXX files.
- **k**: int. Number of centroids. Use `k=4` when generate output.
- **maxIter**: int. Number of iterations as stop criterion. Use `maxIter=10` when generate output.
- Use Euclidean distance as your distance measurement.

Part III. Report Results

Write an additional Hadoop MapReduce program to report top 10 words for each cluster generated by Hadoop. The cluster ID and the words in the output should be tab separated as follow:

```
Cluster0<tab>word00,word01,word02,...,word09
Cluster1<tab>word10,word11,word12,...,word19
...
ClusterN<tab>wordN0,wordN1,wordN2,...,wordN9
```

Submission

1. Mahout script for generating TF-IDF vectors for part I.
2. Hadoop program (.java) for part II & III.
3. The result required in part III.

Option 3: Mahout Recommender System

In this part, you will analyze MovieLens data collected by the GroupLens Research Project at the University of Minnesota. <http://grouplens.org/datasets/movielens/>

TODO

1. Download and unzip the MovieLens 100K Dataset (ml-100k.zip) in the link above.
2. Convert u.data to Mahout readable data

u.data -- The full u data set, 100000 ratings by 943 users on 1682 items. Each user has rated at least 20 movies. Users and items are numbered consecutively from 1. The data is randomly ordered. This is a tab separated list of

user_id<tab>item_id<tab>rating<tab>timestamp.

The time stamps are unix seconds since 1/1/1970 UTC

3. Recommend 10 movies for all the users using Item-Based Collaborative Filtering from the converted dataset and output.

userID<\tab>[itemID1,score1,...,itemID10, score10]
...

Tips

There is a useful tutorial regarding the recommender system with Mahout.

<https://mapr.com/products/mapr-sandbox/hadoop2/tutorial/recommender-tutorial/>

Submission

1. Mahout script (.txt or .sql file)
2. Output file with the required format.

Option 4: Spark/Hadoop Recommender System

Perform the recommender system required in Option with Spark **OR** Hadoop.

A tutorial of spark can be found at <https://spark.apache.org/docs/1.6.0/mllib-collaborative-filtering.html>

Again, you need to output 10 movies for all the users with the following format

userID<\tab>itemID1,itemID2,itemID3 ...,itemID10
...

Submission

1. Spark program (.py)
2. Output file with the required format.