MemoryLeak
Music Mentor User Manual

1. Required Libraries

Apache Mahout

Required Jars:

- *mahout-core-0.9-job.jar*
- *mahout-core-0.9.jar*
- *mahout-integration-0.9.jar*
- *mahout-math-0.9.jar*

Download:
http://ftp.itu.edu.tr/Mirror/Apache/mahout/

Slf4j

Required Jars:

- *slf4j-nop-1.7.7.jar*

Download:
http://www.slf4j.org/download.html

Externalsortinginjava

Required Jars:

- *externalsortinginjava-0.1.9.jar*

Download:
http://repo1.maven.org/maven2/com/google/code/externalsortinginjava/externalsortinginjava/0.1.9/

2. Installation

1. Add the .jar files to the build path of your project.
2. Add the MemoryLeakRecommender-1.0.jar to the build path of your project.
3. Import *Recommender* class, which is in the *Recommender.main* package.
    - `import recommender.main.Recommender;`

3. Methods

- public void train(String folderPath, String workspacePath)

    Process the user logs located in the directory with the given *folderPath* and generates the rating data (ratings.csv) in the directory with the given *workspacePath*. If the directory with *workspacePath* does not exist, it also creates the directory. After processing is done, it trains the recommender with the generated rating data. String represantations of the paths do not matter as long as they are valid paths. The directory with the given *folderPath* can contain only the user logs, directories which contain the user logs and *rating.csv*. *workspacePath* can be the same with the *folderPath*.

- public void loadWorkspace(String workspacePath)

  If the rating data is already created and if the user logs haven't changed, processing the user logs to create a rating data is not necessary. This method trains the recommender with the rating data in the directory given with the *workspacePath*. If *train()* has already called, there is no need to call this method, since it means that *train()* has already trained the recommender with the rating data. String represantations of the path does not matter as long as it is valid path.

- public List<Long> recommend(Long userId)

  Recommends songs to user with given *userId*. To use this function, recommender must be trained with rating data beforehand.

- public List<Long> recommend(Long userId, Long songId)

  Recommends songs to user with give *userId*, as if the user is currently listening to a song with given *songId*. To use this function, recommender must be trained with rating data beforehand.

- public void setPrintOption(boolean opt)

  train() and *loadWorkspace()* methods prints their progresses to inform the user about it. To prevent printing, this method should be called with *false.* To allow printing, this method should be called with *true.* If this method is not called, the default value for printing option of Recommender class is *true*.

4. Example (No puns intended)

Main.java:

```
import recommender.main.Recommender;
import java.util.List;

public class Main
{
     public static void main(String[] args)
     {
             Recommender r = new Recommender(); //Create a recommender instance
             r.train("C:\\Users\\User\\MusicMentor\\logs",
                     "C:/Users/User/MusicMentor/workspace/"); //train the recommender

             //if the rating.csv already exists at C:/Users/User/workspace5/MusicMentorJar/workspace
             //r.loadWorkspace("C:/Users/User/MusicMentor/workspace");

             List<Long> recommended; //list to store recommended song ids

             recommended= r.recommend(2088593L); //recommend songs to the user with user id 2088593
             for(long index : recommended)
                     System.out.println(index); //print recommended song ids

             System.out.println(); //print a newline

             recommended = r.recommend(2088593L, 289720L); //recommend songs to the user with
                                                           //user id 2088593, who is currently
                                                           //listening to the song 289720
             for(long index : recommended)
                     System.out.println(index); //print recommended song ids
     }
}
```

Note: In this example, it is assumed that the folders which contain the user action logs to train the recommender and which have the following paths exist in the user's file system;

- *C:/Users/User/MusicMentor/logs/50*
- *C:/Users/User/MusicMentor/logs/52*

As a result, *train()* is called with the folderPath parameter;

- *C:/Users/User/MusicMentor/logs*