# Mahout and Spark

Instructor: Kunpeng Zhang (kzhang6@uic.edu)
TA: Minghong Xu (mxu29@uic.edu)

**The Deadline: 6:00PM, Mar. 30, 2015 (Monday)**

In this assignment, you need to implement two applications: Item-based collaborative filtering using Mahout and topic modeling LDA using Spark.

**1**. For the item-based collaborative filtering, your input would be the user ID and the user-movie rating CSV file. The output would be top 10 recommended movie names. The following program is user-based collaborative filtering using Mahout.

```
import org.apache.mahout.cf.taste.impl.model.file.*;
import org.apache.mahout.cf.taste.impl.neighborhood.*;
import org.apache.mahout.cf.taste.impl.recommender.*;
import org.apache.mahout.cf.taste.impl.similarity.*;
import org.apache.mahout.cf.taste.model.*;
import org.apache.mahout.cf.taste.neighborhood.*;
import org.apache.mahout.cf.taste.recommender.*;
import org.apache.mahout.cf.taste.similarity.*;
import java.io.*;
import java.util.*;

public class RecommenderTest {
        public static void main(String[] args) throws Exception {
                DataModel model = new FileDataModel(new File("intro.csv"));
                UserSimilarity similarity = new PearsonCorrelationSimilarity(model);
                UserNeighborhood neighborhood = new NearestNUserNeighborhood(5,
                                                        similarity, model);
                Recommender recommender = new GenericUserBasedRecommender(
                                                        model, neighborhood, similarity);
                List<RecommendedItem> recommendations =
                                                        recommender.recommend(1001, 10);
                for (RecommendedItem recommendation : recommendations)
                        System.out.println(recommendation);
        }
}
```

**What you submit for this:**

Your java source code (e.g., RecommenderTest.java) about the item-based collaborative filtering using Mahout

**2**. For the topic modeling LDA, the input would be a path where all documents are stored. There are two types of outputs: For each document, you need to generate top 10 topic probabilities in an descending order; For each topic, you need to give top 10 words with their corresponding probabilities. Your final output should be in the following format:

**Topic-Document Distribution:**

Doc_1 (e.g., file name): topicID-probability, topicID-probability, …, topicID-probability
Doc_2: topicID-probability, topicID-probability, …, topicID-probability
……
Doc_i: topicID-probability, topicID-probability, …, topicID-probability
……
Doc_n: topicID-probability, topicID-probability, …, topicID-probability

**Topic-Term Distribution:**

Topic_1: word-probability, word-probability, ……, word-probability
Topic_2: word-probability, word-probability, ……, word-probability
……..
Topic_10: word-probability, word-probability, ……, word-probability

# Dataset:

The dataset can be downloaded from here: https://archive.ics.uci.edu/ml/machine-learning-databases/bag-of-words/
You need to download docword.nips.txt.gz and vocab.nips.txt.

For each text collection, D is the number of documents, W is the number of words in the vocabulary, and N is the total number of words in the collection (below, NNZ is the number of nonzero counts in the bag-of-words). After tokenization and removal of stopwords, the vocabulary of unique words was truncated by only keeping words that occurred more than ten times. The number of the documents is 1500 and the number of words is 12419.

The format of the docword.*.txt file is 3 header lines, followed by
NNZ triples:
---
D
W
NNZ

docID wordID count
docID wordID count
docID wordID count
docID wordID count
...
docID wordID count
docID wordID count
docID wordID count
---

The format of the vocab.*.txt file is line contains wordID=n.

You need to create 1500 documents first. Document can be named as 1.txt, 2.txt, …, 1500.txt. Then you run your topic-modeling program.

**What you submit for this:**

1) The java source code to generate 1500 documents
2) The java source code of LDA using Spark MLlib package
3) The output file in the format described above
4) Please do not submit 1500 documents!!!

## Requirements:

- Your codes must be readable and clean.
- When you submit your codes through blackboard, you need to put all source codes (.java files, NOT jar files) and some other optional files (e.g., a readme file) into one folder and name that folder as <YOUR UID>_ASSIGN4. **Assignments not following this rule will not be graded**. In addition, no resubmission after TA grades it. Late submission rule: 10% deduction for one day late. Late submission over a week is NOT acceptable.

- **DO NOT copy any codes from others. Otherwise, both will be penalized.**