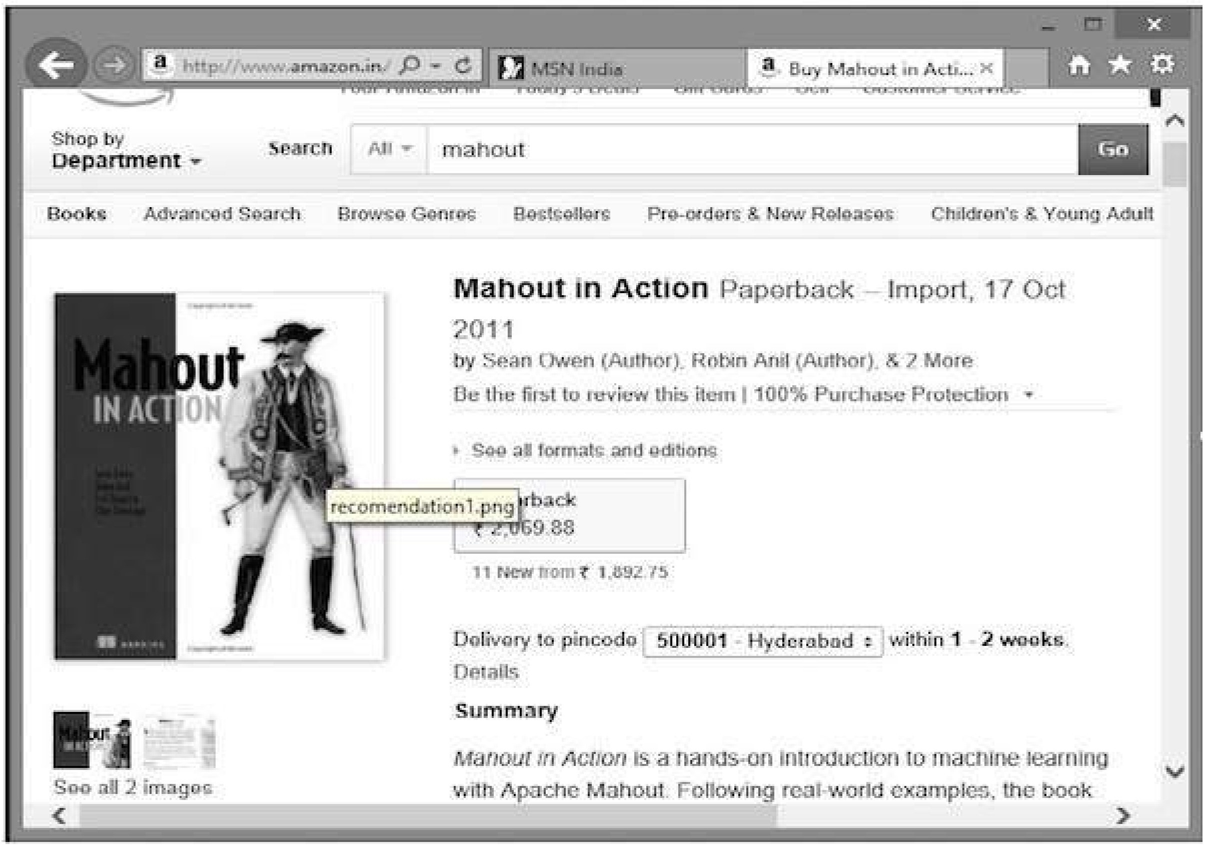
MAHOUT - RECOMMENDATION

This chapter covers the popular machine learning technique called **recommendation,** its mechanisms, and how to write an application implementing Mahout recommendation.

Recommendation

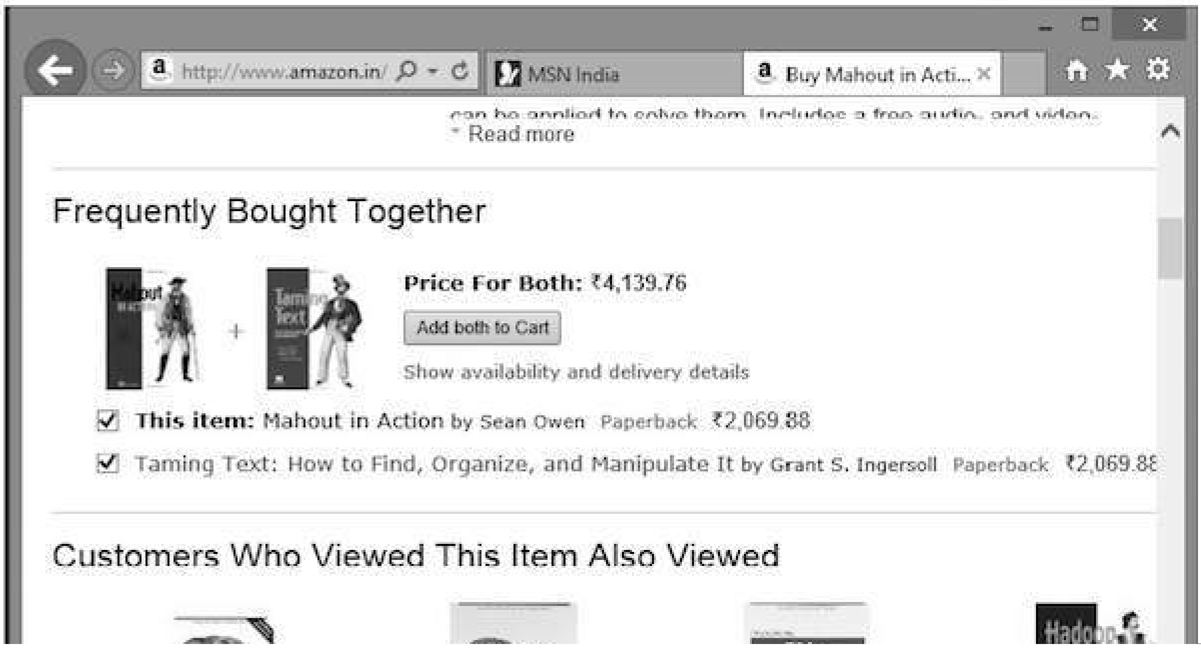
Ever wondered how Amazon comes up with a list of recommended items to draw your attention to a particular product that you might be interested in!

Suppose you want to purchase the book “Mahout in Action” from Amazon:



Mahout in Action Paperback – Import, 17 Oct 2011
by Sean Owen (Author), Robin Anil (Author), & 2 More
Be the first to review this item | 100% Purchase Protection
Paperback ₹ 2,069.88
11 New from ₹ 1,892.75
Delivery to pincode 500001 - Hyderabad within 1 - 2 weeks.
Details
Summary
Mahout in Action is a hands-on introduction to machine learning with Apache Mahout. Following real-world examples, the book

Along with the selected product, Amazon also displays a list of related recommended items, as shown below.



Frequently Bought Together
Price For Both: ₹4,139.76
☑ **This item:** Mahout in Action by Sean Owen Paperback ₹2,069.88
☑ Taming Text: How to Find, Organize, and Manipulate It by Grant S. Ingersoll Paperback ₹2,069.88
Customers Who Viewed This Item Also Viewed

Hadoop: The Definitive Guide
White
(25)
Paperback
₹585.00

Hadoop Operations: A Guide for Developers and ...
Eric Sammer
(4)
Paperback
₹383.00

Programming Pig
Alan Gates
(2)
Paperback
₹375.00

Hadoop in Actio
Chuck Lam
Paperback
₹504.00

Such recommendation lists are produced with the help of **recommender engines**. Mahout provides recommender engines of several types such as:

- user-based recommenders,
- item-based recommenders, and
- several other algorithms.

Mahout Recommender Engine

Mahout has a non-distributed, non-Hadoop-based recommender engine. You should pass a text document having user preferences for items. And the output of this engine would be the estimated preferences of a particular user for other items.

Example

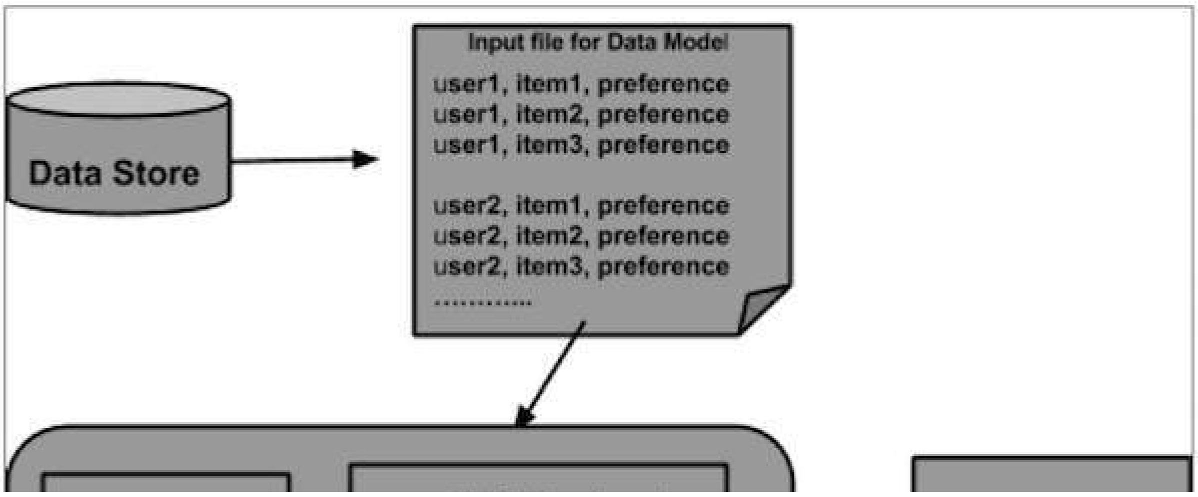
Consider a website that sells consumer goods such as mobiles, gadgets, and their accessories. If we want to implement the features of Mahout in such a site, then we can build a recommender engine. This engine analyzes past purchase data of the users and recommends new products based on that.

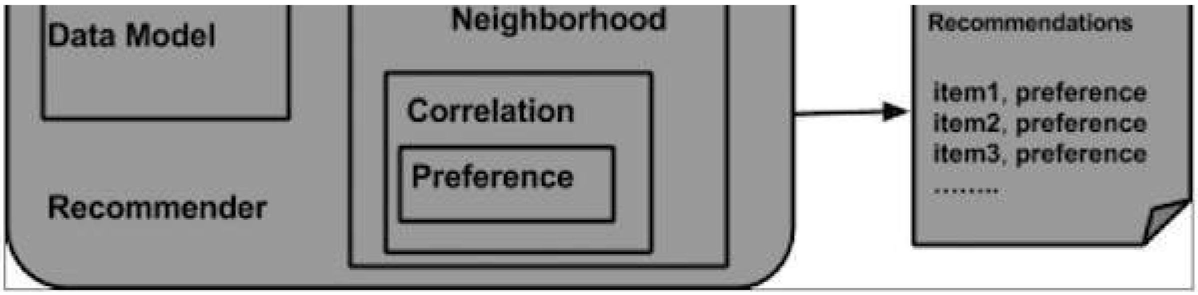
The components provided by Mahout to build a recommender engine are as follows:

- DataModel
- UserSimilarity
- ItemSimilarity
- UserNeighborhood
- Recommender

From the data store, the data model is prepared and is passed as an input to the recommender engine. The Recommender engine generates the recommendations for a particular user. Given below is the architecture of recommender engine.

Architecture of Recommender Engine





Building a Recommender using Mahout

Here are the steps to develop a simple recommender:

Step1: Create DataModel Object

The constructor of **PearsonCorrelationSimilarity** class requires a data model object, which holds a file that contains the Users, Items, and Preferences details of a product. Here is the sample data model file:

```
1,00,1.0
1,01,2.0
1,02,5.0
1,03,5.0
1,04,5.0

2,00,1.0
2,01,2.0
2,05,5.0
2,06,4.5
2,02,5.0

3,01,2.5
3,02,5.0
3,03,4.0
3,04,3.0

4,00,5.0
4,01,5.0
4,02,5.0
4,03,0.0
```

The **DataModel** object requires the file object, which contains the path of the input file. Create the **DataModel** object as shown below.

```
DataModel datamodel = new FileDataModel(new File("input file"));
```

Step2: Create UserSimilarity Object

Create **UserSimilarity** object using **PearsonCorrelationSimilarity** class as shown below:

```
UserSimilarity similarity = new PearsonCorrelationSimilarity(datamodel);
```

Step3: Create UserNeighborhood object

This object computes a "neighborhood" of users like a given user. There are two types of neighborhoods:

- **NearestNUserNeighborhood** - This class computes a neighborhood consisting of the nearest *n* users to a given user. "Nearest" is defined by the given UserSimilarity.

- **ThresholdUserNeighborhood** - This class computes a neighborhood consisting of all the users whose similarity to the given user meets or exceeds a certain threshold. Similarity is defined by the given UserSimilarity.

Here we are using **ThresholdUserNeighborhood** and set the limit of preference to 3.0.

```
UserNeighborhood neighborhood = new ThresholdUserNeighborhood(3.0, similarity, model);
```

Step4: Create Recommender Object

Create **UserbasedRecomender** object. Pass all the above created objects to its constructor as shown below.

```
UserBasedRecommender recommender = new GenericUserBasedRecommender(model,
neighborhood, similarity);
```

Step5: Recommend Items to a User

Recommend products to a user using the recommend method of **Recommender** interface. This method requires two parameters. The first represents the user id of the user to whom we need to send the recommendations, and the second represents the number of recommendations to be sent. Here is the usage of **recommender** method:

```
List<RecommendedItem> recommendations = recommender.recommend(2, 3);

for (RecommendedItem recommendation : recommendations) {
   System.out.println(recommendation);
 }
```

Example Program

Given below is an example program to set recommendation. Prepare the recommendations for the user with user id 2.

```
import java.io.File;
import java.util.List;

import org.apache.mahout.cf.taste.impl.model.file.FileDataModel;
import org.apache.mahout.cf.taste.impl.neighborhood.ThresholdUserNeighborhood;
import org.apache.mahout.cf.taste.impl.recommender.GenericUserBasedRecommender;
import org.apache.mahout.cf.taste.impl.similarity.PearsonCorrelationSimilarity;

import org.apache.mahout.cf.taste.model.DataModel;
import org.apache.mahout.cf.taste.neighborhood.UserNeighborhood;

import org.apache.mahout.cf.taste.recommender.RecommendedItem;
import org.apache.mahout.cf.taste.recommender.UserBasedRecommender;

import org.apache.mahout.cf.taste.similarity.UserSimilarity;

public class Recommender {
   public static void main(String args[]){
      try{
         //Creating data model
         DataModel datamodel = new FileDataModel(new File("data")); //data

         //Creating UserSimilarity object.
         UserSimilarity usersimilarity = new PearsonCorrelationSimilarity(datamodel);

         //Creating UserNeighbourHHood object.
         UserNeighborhood userneighborhood = new ThresholdUserNeighborhood(3.0,
usersimilarity, datamodel);

         //Create UserRecomender
         UserBasedRecommender recommender = new GenericUserBasedRecommender(datamodel,
userneighborhood, usersimilarity);

         List<RecommendedItem> recommendations = recommender.recommend(2, 3);
         for (RecommendedItem recommendation : recommendations) {
            System.out.println(recommendation);
         }

      }catch(Exception e){}

   }
  }
```

Compile the program using the following commands:

```
javac Recommender.java
java Recommender
```

It should produce the following output:

```
RecommendedItem [item:3, value:4.5]
RecommendedItem [item:4, value:4.0]
```