Big Data Science: The art of understanding huge volumes of data

María S. Pérez
E.T.S. de Ingenieros Informáticos
Universidad Politécnica de Madrid
mperez@fi.upm.es
Outline

- Motivation
- Data science: A new paradigm?
- Big Data Challenges
- (Big Data) Science = Big (Data Science)?
- Spark: An alternative to old Hadoop
- Mahout: A framework for Big Data Analysis
Motivation

- IBM: Internet of Things
  - The Internet of Things.mp4

- Concepts:
  - Big Data
  - Data Science
  - Data Deluge
  - Data Mining / Data Analytics
  - Intelligent Management of Data

Source: IBM, Internet of the Things: http://www.youtube.com/watch?v=I5Rba7c6RwQ
Data science: A new paradigm?
Science Evolution

• A thousand years ago, empirical science

• Over the last 500 years, theoretical science

• In the last 50 years, computational science
  – Today most disciplines have both empirical and theoretical branches. A third branch (computational branch) has grown.

• In the last 20 years, data science
  – Computational Science has been evolving to include information management.
  – Scientists are faced with mountains of data

What is Data Science?

• William S. Cleveland, Professor of Computer Science and Statistics, defined Data Science:
  – “Technical areas of data science should be judged by the extent to which they enable the analyst to learn from data”


• A practitioner of data science is called a data scientist, term coined by D.J. Patil and Jeff Hammerbacher.
What is Data Science?

• Data Science is not just data analysis.
• According to Jeffrey Stanton, author of “Introduction to Data Science”, there are four related areas in Data Science:
  – Data architecture: A data scientist would help the system architect by providing input on how the data would need to be routed and organized to support the analysis, visualization and presentation of the data
  – Data acquisition: how the data are collected, and, importantly, how the data are represented prior to analysis and presentation
  – Data analysis: This phase is where data scientists are most heavily involved. There are many technical, mathematical, and statistical aspects to these activities, but the results have to be effectively communicated to the data user.
  – Data archiving: Preservation of collected data in a form that makes it highly reusable (data curation)
What does a data scientist do?

• Amazon’s product recommendation systems
• Google's advertisement valuation systems
• Linkedin's contact recommendation system
• Twitter's trending topics
• Walmart's consumer demand projection systems.

• In common:
  – Large data sets. Not necessarily big data
  – Online or live systems. The data science team deploys a decision procedure or scoring procedure to make decisions or show results to a large number of end users
  – These systems are allowed to make mistakes at some non-negotiable rate
  – None of these systems are concerned with cause. They are successful when they find useful correlations

Source: Practical Data Science with R, Nina Zumel and John Mount, Manning Publications, 2013
Data Scientist Skills

- Evolution from the data analyst role
- Computer science, technology and software engineering methodologies, modeling, statistics, analytics, visualization, databases, machine learning, data mining, big data and maths.
- “Part analyst, part artist” (Renaissance individual).
- Business skills: Influence in making decisions in a business environment (Business intelligence)
- A data scientist does not just collect and report on data, but also evaluates and analyses this data, in order to get informed conclusions which lead to business recommendations
- The data scientist guides a data science project
Data Science Is Multidisciplinary

By Brendan Tierney, 2012
Data Science Venn Diagram

- **Data Hacking Skills**: Manipulate text files at the command-line, understanding vectorized operations, thinking algorithmically.
- **Math & Statistics Knowledge**: Extract insight from data. This requires at least a baseline familiarity with these tools.
- **Substantive Expertise**: Science is about discovery and building knowledge, which requires some motivating questions about the world and hypotheses that can be brought to data and tested with statistical methods.

Data Scientist Skills

• Learning the application domain
• Communicating with data users
• Seeing the big picture of a complex system
• Knowing how data can be represented
• Data transformation and analysis
• Visualization and presentation
• Attention to quality
• Ethical reasoning

Source: Introduction to Data Science, Syracuse University School of Information Studies, J. M. Stanton, May 2012
<table>
<thead>
<tr>
<th>Role</th>
<th>Responsibilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Project Sponsor</td>
<td>Represents the business interests; supports the project.</td>
</tr>
<tr>
<td>The Client</td>
<td>Represents end users' interests; domain expert.</td>
</tr>
<tr>
<td>The Data Scientist</td>
<td>Sets and executes analytic strategy; communicates with Sponsor and Client</td>
</tr>
<tr>
<td>The Data Architect</td>
<td>Manages data and data storage; sometimes manages data collection.</td>
</tr>
<tr>
<td>Operations</td>
<td>Manages infrastructure; deploys final project results.</td>
</tr>
</tbody>
</table>

Source: Practical Data Science with R, Nina Zumel and John Mount, Manning Publications, 2013
## Data Science Project Stages

<table>
<thead>
<tr>
<th>Stage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Define the Goal</td>
<td>Define a measurable and quantifiable goal. <em>What problem I am solving?</em></td>
</tr>
<tr>
<td>Collect &amp; Manage the Data</td>
<td>Identify the data you need, explore it, and condition it to be suitable for analysis. This stage is often the most time-consuming step. <em>What information do I need?</em></td>
</tr>
<tr>
<td>Build the Model</td>
<td>Extract useful insights from the data in order to achieve your goals by using statistics and machine learning.</td>
</tr>
<tr>
<td>Evaluate &amp; Critique the Model</td>
<td>Determine if it meets your goals. <em>Does the model solve my problem?</em></td>
</tr>
<tr>
<td>Present Results &amp; Document</td>
<td>Present the results to your project sponsor and other stakeholders and document the model for the users.</td>
</tr>
<tr>
<td>Deploy the Model</td>
<td>The model is put into operation. Updates if the environment changes.</td>
</tr>
</tbody>
</table>

Source: Practical Data Science with R, Nina Zumel and John Mount, Manning Publications, 2013
Big Data Challenges
Big Data Challenges

• “Our ability to store data is fast overwhelming our ability to process what we store”
  – Disk capacity: from tens of MB in 1980-s to a couple of TB today (several orders of magnitude)
  – Latency: around 2 x improvement in the last quarter century
  – Bandwidth: around 50 x improvement

• Large data problems

Source: “Data-Intensive Text Processing with MapReduce” by J. Lin and C. Dyer
Big Data Challenges. Storage

• HDD:
  – 100 x cheaper than RAM memory
  – 1000 x slower than RAM memory

• SSD (Solid Storage Drive):
  – Many of them are flash-based
    • Seek time is decreased significantly. Faster access time
  – No moving parts
    • They don’t generate significant heat
    • Less energy consumption than HDD
    • Silent
  – SSDs have smaller lifetime than HDD

• Storage Class Memory
  – “A new form of storage created out of flash-based NAND that can provide an intermediate step between high-performance DRAM and cost-effective HDDs. It can provide read performance similar to DRAM and write performance that is significantly faster than HDD technology” (George Crump)
Big Data Challenges. Database

• Relational databases are common in current computing scenarios
  – ACID properties (Atomicity, Consistency, Isolation and Durability)

• Relational databases can’t handle big data size, complexity and the requirements of current apps (thousand of users and millions of queries)

• An alternative database model, called NoSQL database, has appeared
  – They address scalability and performance issues
NoSQL databases

• In some scenarios, it’s possible to lessen the requirements and limitations of conventional databases
• NoSQL databases features:
  – Non-relational
  – Distributed
  – Open source (in general)
  – Horizontally scalable
• The acronym NoSQL was coined in 1998
• Since 2009 there have been an explosion
• They are: schema-free, easy replication support, simple API, eventually consistent / BASE (not ACID), a huge amount of data, no joins, etc.. (read "nosql" as "not only sql")
• http://nosql-database.org
Hadoop

• Open-source implementation called Hadoop (development led by Yahoo and now an Apache project)
• Born in 2007
• Three-side architecture:
  – Main core: Hadoop Common
  – Hadoop Distributed File System: HDFS
  – MapReduce engine: Distributed processing of large data sets on compute clusters
Hadoop

Source: Hadoop in Practice,
http://techannotation.wordpress.com/
2012/09/10/hadoop-in-practice/
Problems of Hadoop

• Scalability:
  – JobTracker runs on a single machine
    • Resource management
    • Job and Task scheduling
    • Monitoring

• Availability and fault tolerance:
  – JobTracker is a single point of failure (if JobTracker fails, all jobs must restart)

• Resource utilization:
  – Predefined number of map and reduce slots

• Limitation in running non-MapReduce applications
  – JobTracker tightly integrated with MapReduce
Evolution of Hadoop

Advantages of YARN

• Better use of the resources
  – No fixed map-reduce slots.
  – YARN provides central resource manager

• YARN can run non-MapReduce applications
  – YARN decouples MapReduce's resource management and scheduling from the data processing
  – For instance, running of interactive querying and streaming data applications simultaneously with MapReduce batch jobs

• YARN is backward compatible

• No more JobTracker and TaskTracker needed
  – 2 separate daemons: Resource Manager and Node Manager
Hadoop ecosystem

Hadoop ecosystem

Accumulo - a sorted, distributed key/value store
http://accumulo.apache.org/

Cassandra - column-oriented database
http://cassandra.apache.org/

Cayenne - object-relational mapping (ORM)
http://cayenne.apache.org/

CouchDB - NoSQL document-oriented datastore
http://couchdb.apache.org/

Gora - in-memory data model & persistence
http://gora.apache.org/

Hadoop - a distributed computing platform:
- HDFS - distributed file system for Hadoop
- MapReduce - parallel computation on clusters
http://hadoop.apache.org/

HBase - column-oriented database on top of Hadoop
http://hbase.apache.org/

Hive - data warehouse with SQL-like access
http://hive.apache.org/

Flume - collection & import of log and event data
http://flume.apache.org/

Lucene - indexing and search
http://lucene.apache.org/

Mahout - library for machine learning & data mining
http://mahout.apache.org/

Pig - high-level programming language for Hadoop
http://pig.apache.org/

Oozie - workflow management for Hadoop
http://oozie.apache.org/

Solr - Lucene-based enterprise search platform
http://lucene.apache.org/solr/

Sqoop - imports data from RDBMS into Hadoop
http://sqoop.apache.org/

Whirr - cloud-agnostic deployment of clusters
http://whirr.apache.org/

Zookeeper - configuration & coordination
http://zookeeper.apache.org/

Hive

- A datawarehouse based on Hadoop, developed by Facebook
- Currently it is an open source project in Hadoop
- Users can write SQL queries, which are transformed into MapReduce in a transparent way
  - SQL-like language called HiveQL
- SQL programmers without experience in MapReduce can use it and integrate it in his/her apps
Apache Pig

• High-level platform for the creation of MapReduce programs
• Apache Pig simplifies the use of Hadoop by allowing SQL-like queries to a distributed dataset
• It uses the language called Pig Latin
  – Instead of writing a MapReduce application, you write a script in Pig Latin that is automatically parallelized and distributed across a cluster
(Big Data) Science = Big (Data Science)?
One of the greatest opportunity in data science is Big Data

- Large volume, heterogeneous, more complex and dynamic relationships among data, difficulty for discovering useful knowledge from Big Data

<table>
<thead>
<tr>
<th>Data Science</th>
<th>Big Data Science</th>
</tr>
</thead>
<tbody>
<tr>
<td>Need of a complete data set</td>
<td>Incomplete data is not a problem</td>
</tr>
<tr>
<td>Data are usually clean</td>
<td>Data are often messy</td>
</tr>
<tr>
<td>Report on what data tells</td>
<td>Exploration to see what data tells me</td>
</tr>
<tr>
<td>Although the dataset can be big, it’s</td>
<td>The size of the dataset is challenging</td>
</tr>
<tr>
<td>manageable</td>
<td></td>
</tr>
</tbody>
</table>

Source: Data Scientist: Hot Big Data Job (Information Week)
Big Data Scientist: An in-demand job

- There is a demand of people telling the CEO what's going to happen next, not what happened last week
- Predictive insights and accurate predictive models are needed in business
- In the era of big data, professionals who know how to use big-data platforms are required
- Both businesses and government agencies look for expertise in data-driven decisions
- Examples:
  - Starbucks use data-intensive analyses to improve stocking, product selection and pricing
  - Harvard Medical School find patterns in clinical data to improve patient diagnoses and treatments
  - Facebook has collected the most extensive dataset on human social behavior. It tries to adapt the platform to this behavior in order to give people the experience they want

Source: Data Science Revealed: A Data-Driven Glimpse into the Burgeoning New Field Introduction (EMC)
Spark: An alternative to Hadoop
Hadoop disadvantages

- MapReduce performs well at one-pass computation, but poorly in multi-pass algorithms
- Not efficient data sharing
  - Intermediate data in distributed file system (HDFS)
  - Slow replication and storage
Hadoop vs Spark Data Sharing

Data Sharing in MapReduce

Data Sharing in Spark

Slow due to replication, serialization, and disk IO

10-100x faster than network and disk

Source: spark-project.org
Spark Goals

• Keep the good features of MapReduce
  – Fault tolerance
  – Data locality
  – Scalability

• Provide distributed memory abstractions to improve the performance

• How?
  – With RDD (Resilient Distributed Datasets)
RDDs

- Immutable collections partitioned and distributed
- They can be rebuilt if a partition is lost
- They can be created by means of data flow operators
- They can be cached for further use
- Parallel operations on RDDs
RDD operations

• Transformations:
  – Map
  – Filter
  – Sample
  – Union
  – GroupByKey
  – ReduceByKey
  – Join
  – Cache
  – ....

• Actions:
  - Reduce
  - Collect
  - Count
  - Save
  - LookupKey
  - ....
Spark advantages

• Simple and efficient programming model by means of making distributed datasets a first-class primitive
• In-memory caching and locality-based parallel operations
• Fault recovery and debugging
Really more than only a layer: BDAS
Some Spark examples

- http://spark.apache.org/examples.html
  - Text search
  - In-memory text search
  - Word count
  - Estimating Pi
Other Big Data Analytics approaches

- IBM System G, a set of graph analytics libraries for Big Data
  - ScaleGraph is an open source version of this tool
- TinkerPop, an open source graph computing framework
- MADlib, a library for scalable in-database analytics, providing SQL-based algorithms for machine learning, data mining and statistics and supporting both structured and unstructured data.
- OpenChorus, a collaborative platform for Data Science, which allows the sharing of Big Data sources, analysis tools, and visualizations.
Mahout: A framework for (Big) Data Analysis
Apache Mahout

• A mahout is a person who rides an elephant
• Hadoop is an elephant

• Mahout:
  – An open source machine learning library from Apache
  – Scalable (Very large data sets)
  – A Java library
Machine learning

- Mahout provides main support to recommender engines (collaborative filtering), clustering, and classification
Recommender engines

• Infer tastes and preferences and recommend items that could be of interest:
  – Amazon.com: Books and other items
  – Netflix: DVDs (2006-2010: $1,000,000 Prize, open competition for the best collaborative filtering algorithm to predict user ratings for films based on previous ratings without any other information)
  – Facebook: identify people who could be of interest
Clustering

• To group items together into clusters that share some similarity
  – Shopping cart
  – Google News groups news articles by topic
  – Yippy.com:
Classification

• Deciding how much a thing is or isn’t part of some type or category
  – Yahoo! Mail: Spam or not spam
  – Google’s Picasa can decide when a region of an image contains a human face.
  – Optical character recognition software classifies scanned text as individual characters
  – Sentiment analysis: identify people’s attitudes and emotional states from language
Other Machine Learning Libraries

• Weka:
  – http://www.cs.waikato.ac.nz/ml/weka

• R:
  – http://www.r-project.org

• What does offer Mahout?
  – Scalability
Scalability

How many photos are uploaded to Flickr every day, month, year?

[Graph showing the number of millions of public photos uploaded per month from January 2004 to December 2013.]

http://www.flickr.com/photos/francemichel/
Mahout Scalability

• Provided by Hadoop:
  – implements the MapReduce paradigm
  – manages storage of the input, intermediate key-value pairs, and output
  – manages partitioning and data transfer between worker machines
  – performs fault tolerance

• Not mandatory the use of Hadoop
Mahout Architecture

- **Three-tiers architecture:**
  - Data storage and shared libraries: Apache Hadoop, utilities, …
  - Algorithms: Recommenders, Clustering, Classifications
  - Applications using Mahout API
Installing Mahout

- Install Apache Maven:
  - Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information
  - http://maven.apache.org

- Install Apache Mahout:
  - http://mahout.apache.org

- Mahout Item Recommender Tutorial using Java and Eclipse (Video tutorial)
  - https://www.youtube.com/watch?v=yD40rVKUwPI

- Install Apache Hadoop:
  - http://hadoop.apache.org
# Mahout Core 1.0-SNAPSHOT API

<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.apache.mahout</td>
<td></td>
</tr>
<tr>
<td>org.apache.mahout.cf.taste.common</td>
<td></td>
</tr>
<tr>
<td>org.apache.mahout.cf.taste.eval</td>
<td></td>
</tr>
<tr>
<td>org.apache.mahout.cf.taste.hadoop</td>
<td></td>
</tr>
<tr>
<td>org.apache.mahout.cf.taste.hadoop.als</td>
<td></td>
</tr>
<tr>
<td>org.apache.mahout.cf.taste.hadoop.item</td>
<td></td>
</tr>
<tr>
<td>org.apache.mahout.cf.taste.hadoop.preparation</td>
<td></td>
</tr>
<tr>
<td>org.apache.mahout.cf.taste.hadoop.similarity.item</td>
<td></td>
</tr>
<tr>
<td>org.apache.mahout.cf.taste.impl.common</td>
<td></td>
</tr>
<tr>
<td>org.apache.mahout.cf.taste.impl.common.jdbc</td>
<td></td>
</tr>
<tr>
<td>org.apache.mahout.cf.taste.impl.eval</td>
<td></td>
</tr>
</tbody>
</table>
Recommenders in Mahout

• Categories of recommender engine algorithms (collaborative filtering):
  – user-based: items liked by people with similar tastes
  – item-based: items which are like the ones you already like

• Knowledge of the properties of the items no required

• Content-based recommendation techniques do require this knowledge.

• Mahout gives directly support to user-based and item-based techniques, and content-based can be built on top of what Mahout provides
My HelloWorld in Mahout

• Mahout contains several kinds of recommender engines
• Steps:
  – Providing the input: data which recommendations are based on (preferences)
  – Creating a recommender
  – Analyzing the output (preferences estimation)
Step 1: Mahout recommender input

• A preference in Mahout:
  – User ID (integer)
  – Item ID (integer)
  – Strength of the user’s preference for the item (number: larger value means stronger positive preferences)

• comma-separated value (csv) format
- What book recommend to user 1?
  - Not 101, 102, or 103 (Discovering new things)
  - Similarity with users 4 and 5:
    - 104, 105 and 106 as possible recommendations
    - The most liked of these possibilities: Item 104
Step 2: Creating a recommender

- Mahout uses several Java interfaces:
  - DataModel interface
  - UserSimilarity interface
  - ItemSimilarity interface
  - UserNeighborhood interface
  - Recommender interface
Step 2: Creating a recommender

class RecommenderIntro {

    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 (2, similarity, model);

        Recommender recommender =
            new GenericUserBasedRecommender (model, neighborhood, similarity);

        List<RecommendedItem> recommendations =
            recommender.recommend(1, 1);

        for (RecommendedItem recommendation : recommendations) {
            System.out.println(recommendation);
        }
    }
}
public class itemRecommend {
    public static void main(String[] args) {
        try {
            model = new FileDataModel(new File("data/intro.csv"));
            UserSimilarity similarity = new PearsonCorrelationSimilarity(model);
            UserNeighborhood neighborhood = new NearestNUserNeighborhood(2, similarity, model);
            Recommender recommender = new GenericUserBasedRecommender(model, neighborhood, similarity);
            List<RecommendedItem> recommendations = recommender.recommend(1, 1);
            for (RecommendedItem recommendation : recommendations) {
                System.out.println(recommendation);
            }
        } catch (IOException e) {
            System.out.println("There was an error");
            e.printStackTrace();
        } catch (TasteException e) {
            System.out.println("There was a Taste error");
            e.printStackTrace();
        }
    }
}
Step 3: Analyzing the output

• Output:
  – RecommendedItem [item:104, value:4.257081]
  – Recommendation: Book 104 to user 1
  – Estimated user 1’s preference for book 104: 4.257081
Evaluation of a recommender

- Goal: producing a score for the recommender
- How?
  - Using a small part of the real dataset as test data
  - Training the recommender with the rest of the dataset and use the test data for measuring its accuracy, comparing the estimated and real values
    - Computing the average difference between estimate and actual preference
    - Root-mean-square of the differences

<table>
<thead>
<tr>
<th></th>
<th>Item 1</th>
<th>Item 2</th>
<th>Item 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual</td>
<td>3.0</td>
<td>5.0</td>
<td>4.0</td>
</tr>
<tr>
<td>Estimate</td>
<td>3.5</td>
<td>2.0</td>
<td>5.0</td>
</tr>
<tr>
<td>Difference</td>
<td>0.5</td>
<td>3.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Average difference</td>
<td>(0.5+3.0+1.0)/3 = 1.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Root-mean-square</td>
<td>√((0.5^2+0.3^2+1.0^2)/3) = 1.8484</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Evaluating the recommender

DataModel model = null;
try { model = new FileDataModel (new File("data/intro.csv"));}
catch (IOException e) {
    System.out.println("There was an error"); e.printStackTrace();
}
RecommenderEvaluator evaluator = new AverageAbsoluteDifferenceRecommenderEvaluator ();
RecommenderBuilder builder = new RecommenderBuilder() {
    @Override
    public Recommender buildRecommender(DataModel model) throws TasteException {
        UserSimilarity similarity = new PearsonCorrelationSimilarity (model);
        UserNeighborhood neighborhood = new NearestNUserNeighborhood (2, similarity, model);
        return new GenericUserBasedRecommender (model, neighborhood, similarity);
    }
};
double score = 0;
try { score = evaluator.evaluate(builder, null, model, 0.7, 1.0);
    catch (TasteException e) {
    System.out.println("There was a Taste error"); e.printStackTrace();
}
System.out.println(score);
}
evaluate() method

```java
public double evaluate(RecommenderBuilder recommenderBuilder,
                      DataModelBuilder dataModelBuilder,
                      DataModel dataModel,
                      double trainingPercentage,
                      double evaluationPercentage)
  throws TasteException
```

Description copied from interface: RecommenderEvaluator

Evaluates the quality of a Recommender's recommendations. The range of values that may be returned depends on the implementation, but lower values mean better recommendations, with 0 being the lowest/best possible evaluation, meaning a perfect match. This method does not accept a Recommender directly, but rather a RecommenderBuilder which can build the Recommender to test on top of a given DataModel.

Implementations will take a certain percentage of the preferences supplied by the given DataModel as "training data". This is typically most of the data, like 90%. This data is used to produce recommendations, and the rest of the data is compared against estimated preference values to see how much the Recommender's predicted preferences match the user's real preferences. Specifically, for each user, this percentage of the user's ratings are used to produce recommendations, and for each user, the remaining preferences are compared against the user's real preferences.

For large datasets, it may be desirable to only evaluate based on a small percentage of the data. The evaluationPercentage controls how many of the DataModel's users are used in evaluation.

To be clear, trainingPercentage and evaluationPercentage are not related. They do not need to add up to 1.0, for example.

Specified by:
evaluate in interface RecommenderEvaluator

Parameters:
- recommenderBuilder - object that can build a Recommender to test
- dataModelBuilder - DataModelBuilder to use, or null, a default DataModel implementation will be used
- dataModel - dataset to use on training
- trainingPercentage - percentage of each user's preferences to use to produce recommendations; the rest are compared to estimated preference values to evaluate Recommender performance
- evaluationPercentage - percentage of users to use in evaluation

Returns:
- a score representing how well the recommender's estimated preferences match real values; lower scores mean a better match and 0 is a perfect match
- throws TasteException - if an error occurs while accessing the DataModel
Precision and recall

- Precision: the proportion of top results that are relevant, considering some definition of relevant for your problem domain
- Recall: the proportion of all relevant results included in the top results

Measuring precision and recall

```java
DataModel model = null;
try { model = new FileDataModel (new File("data/intro.csv"));}
  catch (IOException e) {
      System.out.println("There was an error"); e.printStackTrace();
  }
RecommenderIRStatsEvaluator evaluator = new GenericRecommenderIRStatsEvaluator ();
RecommenderBuilder builder = new RecommenderBuilder() {
    @Override
    public Recommender buildRecommender(DataModel model) throws TasteException {
        UserSimilarity similarity = new PearsonCorrelationSimilarity (model);
        UserNeighborhood neighborhood = new NearestNUserNeighborhood (2, similarity, model);
        return new GenericUserBasedRecommender (model, neighborhood, similarity);
    }
};
try { IRStatistics stats = evaluator.evaluate(builder, null, model, null, 2,
        GenericRecommenderIRStatsEvaluator.CHOOSE_THRESHOLD, 1.0);
      System.out.println(stats.getPrecision()); System.out.println(stats.getRecall());}
  catch (TasteException e) {
      System.out.println("There was a Taste error"); e.printStackTrace();
  }
```
evaluate() method
Analyzing the output

• Precision: 0.75 (on average about three-quarters of recommendations were good)
• Recall: 1 (all good recommendations are among those recommended)
• Problems of precision and recall:
  – They depend on how well a good recommendation can be defined
  – In the previous example, the threshold is defined by the framework
  – A bad choice will affect the results
A 100K example

- http://grouplens.org/datasets/movielens
  - Available rating data sets from the MovieLens web site (http://movielens.org)
  - MovieLens: Free, personalized, non-commercial, ad-free, great movie recommendations

- The 100K dataset:
  - 100,000 ratings (1-5) from 943 users on 1682 movies
  - Each user has rated at least 20 movies
  - Simple demographic info for the users (age, gender, occupation, zip)
  - The data was collected through the MovieLens web site from September 19th, 1997 through April 22nd, 1998.
public class itemRecommend {
    public static void main(String[] args) {
        try {
            DataModel dm = new FileDataModel(new File("data/movies.csv"));
            ItemSimilarity sim = new LogLikelihoodSimilarity(dm);
            GenericItemBasedRecommender recommender = new GenericItemBasedRecommender(dm, sim);
            for (LongPrimitiveIterator items = dm.getItemIDs(); items.hasNext();){
                long itemId = items.nextLong();
                List<RecommendedItem> recommendations = recommender.mostSimilarItems(itemId, 5);
                for (RecommendedItem recommendation: recommendations){
                    System.out.println(itemId +","+ recommendation.getItemID()+ ","+recommendation.getValue());
                }
            }
        } catch (IOException e) {
            System.out.println("There was an error"); e.printStackTrace();
        } catch (TasteException e) {
            System.out.println("There was a Taste error"); e.printStackTrace();
        }
    }
}
Interpreting the results

• Input: 100,000 ratings from 1000 users on 1700 movies

• Output:

<UserID, MovieID, Similarity>
- 1,117,0.9953521
- 1,151,0.9953065
- 1,121,0.9952347
- 1,405,0.99500656
- 1,50,0.99491894
A Big Data example

- From the book “Mahout in action” by S. Owen et al., Manning, 2012

- Wikipedia dataset
  - In 2014, more than 4.5 million articles written in English
    - Extracted and parsed
    - Last update: 30/09/2010
    - links-simple-sorted.zip (323 MB)
    - Associations are one way: a link A->B does not imply B->A
    - Significantly no more items than users or vice versa. No a preference for a user-based or item-based algorithm from a performance point of view
Recommender for this use case

• A link from article A to B: B provides information related to A

• A recommender system:
  – Articles of interest to a reader of A
    • Recommend articles that are pointed to by other articles that also point to some of the same articles that A points to
      – A -> B
      – C -> B
      – C -> D
      – C and D
Scalability issues

- Data:
  - 2 GB of JVM heap space with Mahout
  - Overall heap space around 2.5 GB
- With enough hardware, this could be acceptable
- But, if the input grows to a few billion preferences?
  - Heap requirements top 32 GB
- The size of Wikipedia data set is about the practical upper limit for nondistributed
- Beyond this, a distributed approach is needed
Distributed approach

- Distribution does not imply more efficiency
  - More resources are needed (moving data, computing and storing intermediate results, …)

- But this approach offers the possibility of running computations at scales where non-distributed approaches are not feasible

- Distribution also can complete the work earlier, even if more processing time is needed
  - If DC = 2*NDC
  - 10 CPUs for the computation:
    - Distributed approach 5 times faster than the nondistributed one
Design of a distributed item-based algorithm

• It is necessary to make a variation
  – The nondistributed algorithm is not directly translated to a distributed environment

• Algorithm:
  – 1. Construction of a co-occurrence matrix
  – 2. Computation of user vectors
  – 3. Production of the recommendations
  – 4. Implementation by means of a MapReduce approach
Step 1: Co-occurrence matrix

- Item-based implementations: ItemSimilarity
  - Provides some notion of the degree of similarity between any pair of items

- A square matrix can collect similarity between every pair of items
  - Symmetric across the diagonal
  - It’s not a user-item matrix.

- Instead of computing the similarity between each pair of items, we’ll compute the times each pair of items occurs in some user preferences (co-occurrence matrix)
  - The co-occurrence matrix plays a similar role to ItemSimilarity, but it can be used in a distributed approach
Step 2: User vectors

- Data model with n items
  - A vector with one dimension for each item
  - No preference: 0 value
  - This vector is usually quite spare (many 0’s), because users usually show a preference for a small subset of items
Step 3: Production of the recommendations

- Multiply the co-occurrence matrix with the user vector

<table>
<thead>
<tr>
<th></th>
<th>101</th>
<th>102</th>
<th>103</th>
<th>104</th>
<th>105</th>
<th>106</th>
<th>107</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>102</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>103</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>104</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>105</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>106</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>107</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>U3</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>R</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>40.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

From the book: “Mahout in action” by S. Owen et al., Manning, 2012
Step 4: MapReduce implementation

- Features of this use case that make it suitable for MapReduce:
  - The computation is feasible to be divided in pieces:
    - Creating user vectors, counting co-occurrences and computing the recommendation vector

- MapReduce:
  - Input: (K1,V1) pairs, usually as HDFS files
  - map(K1,V1): 0 or more (K2,V2)
  - All V2 for each K2 are combined
  - reduce(K2, V2): 0 or more (K3, V3) (HDFS)

- Source code available in: http://www.manning.com/lam (Chapter 6)
  - Implementation of Hadoop’s MapReduce Mapper and Reducer Java interfaces
MapReduce stage 1

• Generating user vectors (\textit{WikipediaToItemPrefsMapper} and \textit{WikipediaToUserVectorReducer} classes)

• Input: (Long, string) pairs
  – Long value: position in the file
  – String value: userId: itemID1, itemID2, itemID3, ..
  – Example: 239/ 98955: 590 22 9059

• Map function: emits new key-value pairs: user ID mapped to item ID
  – Example: 98955 / 590

• Reduce function: builds a Vector from all item IDs for the user, with a user ID mapped to the user’s preference vector (values in the vector are 0 or 1)
  – Example: 98955 / [590: 1.0, 22:1.0, 9059: 1.0]
MapReduce stage 2

• Computing co-occurrence (*UserVectorToCooccurrenceMapper* and *UserVectorToCooccurrenceReducer* classes)

• Input: userIds/ Vector of user preference pairs
  – The output of MapReduce stage 1
  – Example: 98955 / [590: 1.0, 22:1.0, 9059: 1.0]

• Map function: determines the co-occurrences and emits one pair of item IDs for each co-occurrence (item ID to item ID). Both mappings are recorded (item ID1 /item ID2 and itemID2 / itemID1)
  – Example: 590 / 22

• Reduce function: counts for each item ID all co-occurrences. These can be used as rows or columns of the co-occurrence matrix
  – Example: 590 / [22: 3.0, 95: 1.0, ….9059:1.0, ….]
MapReduce stage 3

- Previous to computing the recommendation vector, it is necessary to make some intermediary computation: `CooccurrenceColumnWrapperMapper` and `UserVectorSplitterMapper`
- No Reducer in this stage (mapper are run separately and the output is passed through a no-op Reducer and saved in two locations)
- This is due to the fact that two different kinds of data are joined in one computation: co-occurrence column vectors and user preference values. Hadoop does not allow to have different types in a reducer
MapReduce stage 4

- Computing recommendation vector (*PartialMultiplyMapper* and *AggregateAndRecommendReducer* classes)
- Input: co-occurrence matrix columns/ user preference by items
  - Example: 590 / [22: 3.0, 95: 1.0, ….9059:1.0, ….] and 590/ [98955:1.0]
- Map function: outputs the co-occurrence column for each associated user times the preference value
  - Example: 590 / [22: 3.0, 95: 1.0, ….9059:1.0, ….]
- Reduce function: sums all the vectors, giving the user’s final recommendation vector
  - Example: 590 / [22: 4.0, 45:3.0, 95: 11.0, ….9059:1.0, ….]
- A previous optimization is made: a combiner, which is a miniature reducer operation to save I/O: *AggregateCombiner*
Running with Hadoop

• Some resources:
  – Hadoop documentation:
    • http://hadoop.apache.org/docs/current/
  – Single node cluster:
  – Hadoop Tutorial:
    • https://developer.yahoo.com/hadoop/tutorial
Running with Hadoop

• The class used for linking all together is org.apache.mahout.cf.taste.hadoop.item.RecommenderJob (from the Mahout source distribution; mahout-core-x.y.jar)

• Combine the mappers and reducers in a JAR file (by using Maven)

• Run the command:
  – hadoop jar <nameJARfile>
    org.apache.mahout.cf.taste.hadoop.item.RecommenderJob <options>
A further step

- If you don’t have a large cluster, run it in a public cloud:
  - Amazon Elastic MapReduce (Amazon EMR)
    - https://aws.amazon.com/es/elasticmapreduce
  - Step by step:
References
References

- EMC, Data Science Revealed: A Data-Driven Glimpse into the Burgeoning New Field Introduction
- J. Lin et al., “Data-Intensive Text Processing with MapReduce”, 2010
- J. M. Stanton, Introduction to Data Science, Syracuse University School of Information Studies, May 2012
- Nina Zumel and John Mount, Practical Data Science with R, Manning Publications, 2013
Big Data Science: The art of understanding huge volumes of data

María S. Pérez
E.T.S. de Ingenieros Informáticos
Universidad Politécnica de Madrid
mperez@fi.upm.es