SAMPLE CHAPTER

Mahout IN ACTION

Sean Owen
Robin Anil
Ted Dunning
Ellen Friedman

MANNING

Requires Adobe Acrobat Reader to play audio and video links

Mahout in Action

by Sean Owen
Robin Anil
Ted Dunning
Ellen Friedman

Chapter 1

brief contents

1 Meet Apache Mahout

This chapter covers

- What Apache Mahout is, and where it came from
- A glimpse of recommender engines, clustering, and classification in the real world
- Setting up Mahout

As you may have guessed from the title, this book is about putting a particular tool, Apache Mahout, to effective use in real life. It has three defining qualities.

First, Mahout is an open source *machine learning* library from Apache. The algorithms it implements fall under the broad umbrella of *machine learning* or *collective intelligence*. This can mean many things, but at the moment for Mahout it means primarily recommender engines (collaborative filtering), clustering, and classification.

It's also *scalable*. Mahout aims to be the machine learning tool of choice when the collection of data to be processed is very large, perhaps far too large for a single machine. In its current incarnation, these scalable machine learning implementations in Mahout are written in Java, and some portions are built upon Apache's Hadoop distributed computation project.

Finally, it's a *Java library*. It doesn't provide a user interface, a prepackaged server, or an installer. It's a framework of tools intended to be used and adapted by developers.

To set the stage, this chapter will take a brief look at the sorts of machine learning that Mahout can help you perform on your data—using recommender engines, clustering, and classification—by looking at some familiar real-world instances.

In preparation for hands-on interaction with Mahout throughout the book, you'll also step through some necessary setup and installation.

1.1 Mahout's story

First, some background on Mahout itself is in order. You may be wondering how to pronounce *Mahout*: in the way it's commonly Anglicized, it should rhyme with *trout*. It's a Hindi word that refers to an elephant driver, and to explain that one, here's a little history.

Mahout began life in 2008 as a subproject of Apache's Lucene project, which provides the well-known open source search engine of the same name. Lucene provides advanced implementations of search, text mining, and information-retrieval techniques. In the universe of computer science, these concepts are adjacent to machine learning techniques like clustering and, to an extent, classification. As a result, some of the work of the Lucene committers that fell more into these machine learning areas was spun off into its own subproject. Soon after, Mahout absorbed the Taste open source collaborative filtering project.

No. 1

Figure 1.1 shows some of Mahout's lineage within the Apache Software Foundation. As of April 2010, Mahout became a top-level Apache project in its own right, and got a brand-new elephant rider logo to boot.

Much of Mahout's work has been not only implementing these algorithms conventionally, in an efficient and scalable way, but also converting some of these algorithms to work at scale on top of Hadoop. Hadoop's mascot is an elephant, which at last explains the project name!

Mahout incubates a number of techniques and algorithms, many still in development or in an experimental phase (https://cwiki.apache.org/confluence/display/MAHOUT/Algorithms). At this early stage in the project's life, three core themes are evident: recommender engines (collaborative filtering), clustering, and classification. This is by no means all that exists within Mahout, but they are the most prominent and mature themes at the time of writing. These, therefore, are the focus of this book.

Chances are that if you're reading this, you're already aware of the interesting potential of these three families of techniques. But just in case, read on.

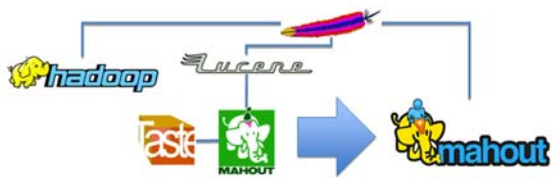

Figure 1.1 Apache Mahout and its related projects within the Apache Software Foundation

1.2 Mahout's machine learning themes

Although Mahout is, in theory, a project open to implementations of all kinds of machine learning techniques, it's in practice a project that focuses on three key areas of machine learning at the moment. These are recommender engines (collaborative filtering), clustering, and classification.

1.2.1 Recommender engines

Recommender engines are the most immediately recognizable machine learning technique in use today. You'll have seen services or sites that attempt to recommend books or movies or articles based on your past actions. They try to infer tastes and preferences and identify unknown items that are of interest:

- Amazon.com is perhaps the most famous e-commerce site to deploy recommendations. Based on purchases and site activity, Amazon recommends books and other items likely to be of interest. See figure 1.2.
- Netflix similarly recommends DVDs that may be of interest, and famously offered a $1,000,000 prize to researchers who could improve the quality of their recommendations.
- Dating sites like Líbímseti (discussed later) can even recommend people to people.
- Social networking sites like Facebook use variants on recommender techniques to identify people most likely to be as-yet-unconnected friends.

As Amazon and others have demonstrated, recommenders can have concrete commercial value by enabling smart cross-selling opportunities. One firm reports that recommending products to users can drive an 8 to 12 percent increase in sales.[1]

1.2.2 Clustering

Clustering is less apparent, but it turns up in equally well-known contexts. As its name implies, clustering techniques attempt to group a large number of things together into clusters that share some similarity. It's a way to discover hierarchy and order in a

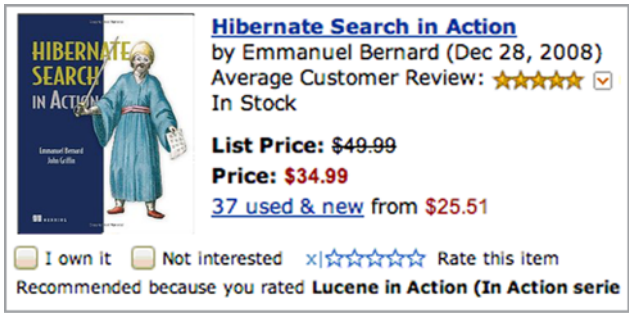

Figure 1.2 A recommendation from Amazon. Based on past purchase history and other activity of customers like the user, Amazon considers this to be something the user is interested in. It can even list similar items that the user has bought or liked that in part caused the recommendation.

[1] Practical eCommerce, "10 Questions on Product Recommendations," http://mng.bz/b6A5

Figure 1.3 A sample news grouping from Google News. A detailed snippet from one representative story is displayed, and links to a few other similar stories within the cluster for this topic are shown. Links to all the stories that are clustered together in this topic are available too.

large or hard-to-understand data set, and in that way reveal interesting patterns or make the data set easier to comprehend.

- Google News groups news articles by topic using clustering techniques, in order to present news grouped by logical story, rather than presenting a raw listing of all articles. Figure 1.3 illustrates this.
- Search engines like Clusty group their search results for similar reasons.
- Consumers may be grouped into segments (clusters) using clustering techniques based on attributes like income, location, and buying habits.

Clustering helps identify structure, and even hierarchy, among a large collection of things that may be otherwise difficult to make sense of. Enterprises might use this technique to discover hidden groupings among users, or to organize a large collection of documents sensibly, or to discover common usage patterns for a site based on logs.

1.2.3 Classification

Classification techniques decide how much a thing is or isn't part of some type or category, or how much it does or doesn't have some attribute. Classification, like clustering, is ubiquitous, but it's even more behind the scenes. Often these systems learn by reviewing many instances of items in the categories in order to deduce classification rules. This general idea has many applications:

- Yahoo! Mail decides whether or not incoming messages are spam based on prior emails and spam reports from users, as well as on characteristics of the email itself. A few messages classified as spam are shown in figure 1.4.
- Google's Picasa and other photo-management applications can decide when a region of an image contains a human face.
- Optical character recognition software classifies small regions of scanned text into individual characters.
- Apple's Genius feature in iTunes reportedly uses classification to classify songs into potential playlists for users.

Classification helps decide whether a new input or thing matches a previously observed pattern or not, and it's often used to classify behavior or patterns as unusual. It could be used to detect suspicious network activity or fraud. It might be used to figure out when a user's message indicates frustration or satisfaction.

Figure 1.4 Spam messages as detected by Yahoo! Mail. Based on reports of email spam from users, plus other analysis, the system has learned certain attributes that usually identify spam. For example, messages mentioning "Viagra" are frequently spam—as are those with clever misspellings like "v1agra." The presence of such terms is an example of an attribute that a spam classifier can learn.

Each of these techniques works best when provided with a large amount of good input data. In some cases, these techniques must not only work on large amounts of input, but must produce results quickly, and these factors make scalability a major issue. And, as mentioned before, one of Mahout's key reasons for being is to produce implementations of these techniques that do scale up to huge input.

1.3 Tackling large scale with Mahout and Hadoop

How real is the problem of scale in machine learning algorithms? Let's consider the size of a few problems where you might deploy Mahout.

Consider that Picasa may have hosted over half a billion photos even three years ago, according to some crude estimates.[2] This implies millions of new photos per day that must be analyzed. The analysis of one photo by itself isn't a large problem, even though it's repeated millions of times. But the learning phase can require information from each of the billions of photos simultaneously—a computation on a scale that isn't feasible for a single machine.

According to a similar analysis, Google News sees about 3.5 million new news articles *per day*. Although this does not seem like a large amount in absolute terms, consider that these articles must be clustered, along with other recent articles, in *minutes* in order to become available in a timely manner.

The subset of rating data that Netflix published for the Netflix Prize contained 100 million ratings. Because this was just the data released for contest purposes, presumably the total amount of data that Netflix actually has and must process to create recommendations is many times larger!

Machine learning techniques must be deployed in contexts like these, where the amount of input is large—so large that it isn't feasible to process it all on one computer, even a powerful one. Without an implementation such as Mahout, these would be impossible tasks. This is why Mahout makes scalability a top priority, and why this book will focus, in a way that others don't, on dealing with large data sets effectively.

Sophisticated machine learning techniques, applied at scale, were until recently only something that large, advanced technology companies could consider using. But today computing power is cheaper than ever and more accessible via open source frameworks like Apache's Hadoop. Mahout attempts to complete the puzzle by

[2] *Google Blogoscoped*, "Overall Number of Picasa Photos" (March 12, 2007), http://blogoscoped.com/archive/2007-03-12-n67.html

providing quality, open source implementations capable of solving problems at this scale with Hadoop, and putting this into the hands of all technology organizations.

Some of Mahout makes use of Hadoop, which includes an open source, Java-based implementation of the MapReduce distributed computing framework popularized and used internally at Google (http://labs.google.com/papers/mapreduce.html). MapReduce is a programming paradigm that at first sounds odd, or too simple to be powerful. The MapReduce paradigm applies to problems where the input is a set of key-value pairs. A *map* function turns these key-value pairs into other intermediate key-value pairs. A *reduce* function merges in some way all values for each intermediate key to produce output. Actually, many problems can be framed as MapReduce problems, or as a series of them. The paradigm also lends itself quite well to parallelization: all of the processing is independent and so can be split across many machines. Rather than reproduce a full explanation of MapReduce here, we refer you to tutorials such as the one provided by Hadoop (http://hadoop.apache.org/mapreduce/docs/current/mapred_tutorial.html).

Hadoop implements the MapReduce paradigm, which is no small feat, even given how simple MapReduce sounds. It manages storage of the input, intermediate key-value pairs, and output; this data could potentially be massive and must be available to many worker machines, not just stored locally on one. It also manages partitioning and data transfer between worker machines, as well as detection of and recovery from individual machine failures. Understanding how much work goes on behind the scenes will help prepare you for how relatively complex using Hadoop can seem. It's not just a library you add to your project. It's several components, each with libraries and (several) standalone server processes, which might be run on several machines. Operating processes based on Hadoop isn't simple, but investing in a scalable, distributed implementation can pay dividends later: your data may quickly grow to great size, and this sort of scalable implementation is a way to future-proof your application.

In chapter 6, this book will try to cut through some of that complexity to get you running on Hadoop quickly, after which you can explore the finer points and details of operating full clusters and tuning the framework. Because this complex framework that needs a great deal of computing power is becoming so popular, it's not surprising that cloud computing providers are beginning to offer Hadoop-related services. For example, Amazon offers Elastic MapReduce (http://aws.amazon.com/elasticmapreduce/), a service that manages a Hadoop cluster, provides the computing power, and puts a friendlier interface on the otherwise complex task of operating and monitoring a large-scale job with Hadoop.

1.4 Setting up Mahout

You'll need to assemble some tools before you can play along at home with the code we'll present in the coming chapters. We assume you're comfortable with Java development already.

Mahout and its associated frameworks are Java-based and therefore platform-independent, so you should be able to use it with any platform that can run a modern JVM. At times, we'll need to give examples or instructions that will vary from platform to platform. In particular, command-line commands are somewhat different in a Windows shell than in a FreeBSD tcsh shell. We'll use commands and syntax that work with bash, a shell found on most Unix-like platforms. This is the default on most Linux distributions, Mac OS X, many Unix variants, and Cygwin (a popular Unix-like environment for Windows). Windows users who wish to use the Windows shell are the most likely to be inconvenienced by this. Still, it should be simple to interpret and translate the listings given in this book to work for that shell.

1.4.1 Java and IDEs

Java is likely already installed on your personal computer if you've done any Java development so far. Note that Mahout requires Java 6. If you're not sure which Java version you have, open a terminal and type `java -version`. If the reported version doesn't begin with 1.6, you need to also install Java 6.

Windows and Linux users can find a Java 6 JVM from Oracle at http://www.oracle.com/technetwork/java/. Apple provides a Java 6 JVM for Mac OS X 10.5 and 10.6. In Mac OS X, if it doesn't appear that Java 6 is being used, open the Java Preferences application under the /Applications/Utilities folder. This will allow you to select Java 6 as the default.

Most people will find it quite a bit easier to edit, compile, and run this book's examples with the help of an IDE; this is *strongly* recommended. Eclipse (http://www.eclipse.org) is the most popular, free Java IDE. Installing and configuring Eclipse is beyond the scope of this book, but you should spend some time becoming familiar with it before proceeding. NetBeans (http://netbeans.org/) is also a popular, free IDE. IntelliJ IDEA (http://www.jetbrains.com/idea/index.html) is another powerful and popular IDE, and a free community version is now available.

As an example of what you can do with an IDE, IDEA can create a new project from an existing Maven model; if you specify the root directory of the Mahout source code upon creating a project, it will automatically configure and present the entire project in an organized manner. It's then possible to drop the source code found throughout this book into the examples/src/main/java/ source root and run it from within IDEA with one click—the details of dependencies and compilation are managed automatically. This should prove far easier than attempting to compile and run the code manually.

> **NOTE** If the test program uses a file with input data, it usually should run in the same directory as the file with the data. Consult your IDE's guide for details about how to set up a working directory for each of the examples.

1.4.2 Installing Maven

As with many Apache projects, Mahout's build and release system is built around Maven (http://maven.apache.org). Maven is a command-line tool that manages dependencies, compiles code, packages releases, generates documentation, and publishes formal releases. Although it has some superficial resemblance to the also-popular Ant build tool, it isn't the same. Ant is a flexible, lower-level scripting language, and Maven is a higher-level tool more purpose-built for dependency and release management. Because Mahout uses Maven, you should install Maven yourself.

Mac OS X users will be pleased to find that Maven should already be installed. If not, install Apple's Developer Tools. Type `mvn --version` on the command line. If you successfully see a version number, and the version is at least 2.2, you're ready to go. If not, you should install a local copy of Maven.

Users of Linux distributions with a decent package management system may be able to use it to quickly obtain a recent version of Maven. Otherwise, standard procedure would be to download a binary distribution, unpack it to a common location, such as /usr/local/maven, and then edit bash's configuration file, ~/.bashrc, to include a line like `export PATH=/usr/local/maven/bin:$PATH`. This will ensure that the `mvn` command is always available.

If you're using an IDE like Eclipse or IntelliJ, it already includes Maven integration. Refer to its documentation to learn how to enable the Maven integration. This will make working with Mahout in an IDE much simpler, as the IDE can use the project's Maven configuration file (pom.xml) to instantly configure and import the project.

> **NOTE** For Eclipse, you'll need to install the m2eclipse plugin (http://www.eclipse.org/m2e/). For NetBeans, Maven support is available out of the box starting with version 6.7; for previous versions, you'll need to install a separate plugin.

1.4.3 Installing Mahout

Mahout is still in development, and this book was written to work with the 0.5 release of Mahout. This release and others may be downloaded by following the instructions at https://cwiki.apache.org/confluence/display/MAHOUT/Downloads; the archive of the source code may be unpacked anywhere that's convenient on your computer.

Because Mahout is changing frequently, and bug fixes and improvements are added regularly, it may be useful to use a later release than 0.5 (or even the latest unreleased code from Subversion; see https://cwiki.apache.org/confluence/display/MAHOUT/Version+Control). Future point releases should be backwards compatible with the examples in this book.

Once you've obtained the source, either from Subversion or from a release archive, create a new project for Mahout in your IDE. This is IDE-specific; refer to its documentation for particulars of how this can be accomplished. It will be easiest to use your IDE's Maven integration to import the Maven project from the pom.xml file in the root of the project source.

Once all of these setup steps are completed, you can easily create a new source directory within this project to hold the sample code that will be introduced in upcoming chapters. With the project properly configured, you should be able to compile and run the code transparently with no further effort.

The source code for the examples is available from Manning's site (http://www.manning.com/MahoutinAction/) or from GitHub (https://github.com/tdunning/MiA). Use the instructions provided with the source code to set up your work environment.

1.4.4 Installing Hadoop

For some activities later in this book, you'll need your own local installation of Hadoop. You don't need a cluster of computers to run Hadoop. Setting up Hadoop isn't difficult, but it's not trivial. Rather than repeat the procedures, we'll direct you to obtain a copy of Hadoop version 0.20.2 from the Hadoop website at http://hadoop.apache.org/common/releases.html, and then set up Hadoop for pseudo-distributed operation by following the Single Node Setup documentation (http://hadoop.apache.org/common/docs/current/single_node_setup.html).

1.5 Summary

Mahout is a young, open source, scalable machine learning library from Apache, and this book is a practical guide to using Mahout to solve real problems with machine learning techniques. In particular, you'll soon explore recommender engines, clustering, and classification. If you're a researcher familiar with machine learning theory and you're looking for a practical how-to guide, or you're a developer looking to quickly learn best practices from practitioners, this book is for you.

These techniques are no longer merely theory. We've already noted some well-known examples of recommender engines, clustering, and classification deployed in the real world: e-commerce, email, videos, photos, and more involve large-scale machine learning. These techniques have been deployed to solve real problems and even generate value for enterprises—and they're now accessible via Mahout.

We've also noted the vast amount of data sometimes employed with these techniques—scalability is a uniquely persistent concern in this area. We took a first look at MapReduce and Hadoop and how they power some of the scalability that Mahout provides.

Because this will be a hands-on, practical book, we've set you up to begin working with Mahout right away. At this point, you should have assembled the tools you'll need to work with Mahout and be ready for action. Because this book is intended to be practical, let's wrap up the opening remarks now and get on to some real code with Mahout. Read on!

JAVA/MACHINE LEARNING

Mahout IN ACTION

Owen • Anil • Dunning • Friedman

Free eBook
SEE INSERT

A computer system that learns and adapts as it collects data can be really powerful. Mahout, Apache's open source machine learning project, captures the core algorithms of recommendation systems, classification, and clustering in ready-to-use, scalable libraries. With Mahout, you can immediately apply to your own projects the machine learning techniques that drive Amazon, Netflix, and others.

This book covers machine learning using Apache Mahout. Based on experience with real-world applications, it introduces practical use cases and illustrates how Mahout can be applied to solve them. It places particular focus on issues of scalability and how to apply these techniques against large data sets using the Apache Hadoop framework.

What's Inside

- Use group data to make individual recommendations
- Find logical clusters within your data
- Filter and refine with on-the-fly classification
- Free audio and video extras

This book is written for developers familiar with Java—no prior experience with Mahout is assumed.

Sean Owen helped build Google's Mobile Web search and launched the Taste framework, now part of Mahout. **Robin Anil** contributed the Bayes classifier and frequent pattern mining implementations to Mahout. **Ted Dunning** contributed to the Mahout clustering, classification, and matrix decomposition algorithms. **Ellen Friedman** is an experienced writer with a doctorate in biochemistry.

For access to the book's forum and a free audio- and video-enhanced ebook for owners of this book, go to manning.com/MahoutinAction

"A hands-on discussion of machine learning with Mahout."
—Isabel Drost
Cofounder Apache Mahout

"The writing makes a complex topic easy to understand."
—Rick Wagner, Red Hat

"Essential Mahout, authored by the core developer team."
—Philipp K. Janert
author of *Gnuplot in Action*

"Dramatically reduces the learning curve."
—David Grossman
Illinois Institute of Technology

"Recommendations, clustering, and classification all lucidly explained."
—John S. Griffin, Overstock.com

ISBN-13: 978-1-935182-68-9
ISBN-10: 1-935182-68-4
54499
9 781935 182689

MANNING

$44.99 / Can $47.99 [INCLUDING eBOOK]