Lecture 11

Predictive Analytics with Mahout

Zoran B. Djordjević

Reference

- These slides follow to a good degree
  Chapter 9 of “Hadoop in Practice” by Alex Holmes, Manning 2012

Predictive Analytics

- Predictive analytics is the field of deriving information about processes in future or values on new points in parameter space based on current and historical data.
- Predictive Analytics examines large datasets (*big data*) and derives meaningful insights from that data, optimally in the form of new values for new parameters.
- Predictive analytics can be broken down into three broad categories:
  - ***Recommender systems*** which suggest items based on the past behavior or interest. These items could, for example, be users in a social network, or products and services on retail websites.
  - ***Classification* systems** (known as *supervised learning systems*) infer or assign a category to previously unseen data, based on prior observations about similar data. Examples of classification include email spam filtering and detection of fraudulent credit card transactions.
  - **Clustering systems** (known as *unsupervised learning systems*) group data together into clusters. They do so without learning the characteristics about related data. Clustering is useful when we are trying to discover hidden structures in our data, such as user habits. Clustering is frequently done as preparation for classification.

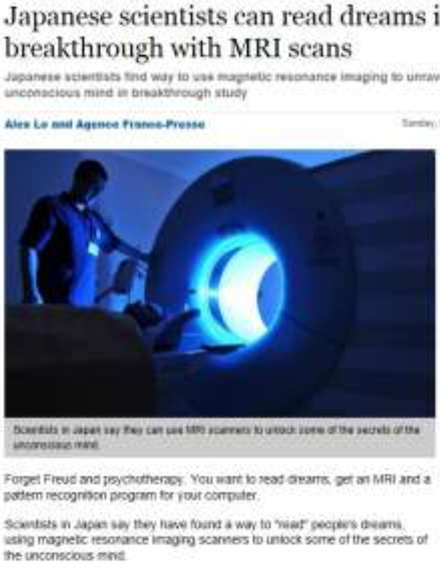
Mahout

- Mahout is a machine learning library which includes implementations of those three classes of predictive analytics techniques.
- Many of Mahout algorithms have Map Reduce implementations, and this is where Mahout comes into its own—its ability to work with huge datasets that other predictive analytics tools can't support.
- Mahout only starts to make sense if you're working with data sets that number in the millions or more.
- Mahout is an Apache Software Foundation project with the prime objective to create scalable machine learning libraries under the Apache Software License.
- It appears that recently, Mahout is moving away from Map Reduce techniques. It remains machine learning library.

Let us Read Dreams, AAAS Science

- In a study, published in the journal Science, on April 4th, 2013, researchers at the ATR Computational Neuroscience Laboratories, Kyoto, Japan, reported on use of magnetic resonance imaging (MRI) scans to locate which part of the brain was active during the first moments of sleep.
- After the recording, scientists would wake up the dreamers and asked them what images they had seen. The process was repeated several 100 times.
- These answers were compared with the brain maps that had been produced by the MRI scanner
- On subsequent scans Japanese researchers were able to predict dreams the volunteers had with an accuracy of 60 percent. The accuracy increased to more than 70 percent with 15 specific items including men, words and books.
- "We have concluded that we successfully decoded dreams with a distinctively high success rate," said Yukiyasu Kamitani, a senior researcher at the laboratories and head of the study team.

Dreams come in Technicolor



Japanese scientists can read dreams in breakthrough with MRI scans

Japanese scientists find way to use magnetic resonance imaging to unravel nighttim unconscious mind in breakthrough study

Alex Lo and Agence France-Presse

Sunday, 07 April 2013

Scientists in Japan say they can use MRI scanners to unlock some of the secrets of the unconscious mind

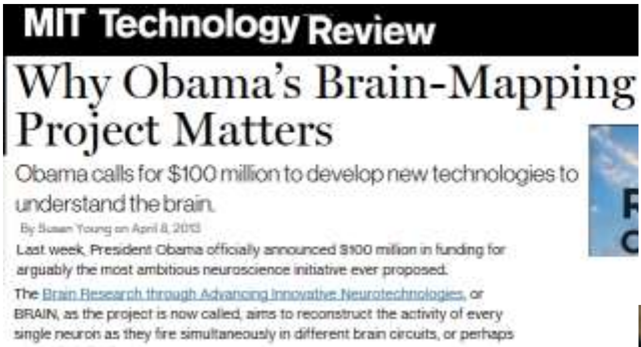
Forget Freud and psychotherapy. You want to read dreams, get an MRI and a pattern recognition program for your computer.

Scientists in Japan say they have found a way to "read" people's dreams, using magnetic resonance imaging scanners to unlock some of the secrets of the unconscious mind.

RELAT

Mapping

Brain Research Through Advancing Initiative Neurotechnologies



MIT Technology Review

Why Obama's Brain-Mapping Project Matters

Obama calls for $100 million to develop new technologies to understand the brain.

By Susan Young on April 8, 2013

Last week, President Obama officially announced $100 million in funding for arguably the most ambitious neuroscience initiative ever proposed.

The Brain Research through Advancing Innovative Neurotechnologies, or BRAIN, as the project is now called, aims to reconstruct the activity of every single neuron as they fire simultaneously in different brain circuits, or perhaps

B.R.A.I.N

- If they ask you for your thoughts, which technique would you prefer?

Waterboarding

Recommenders

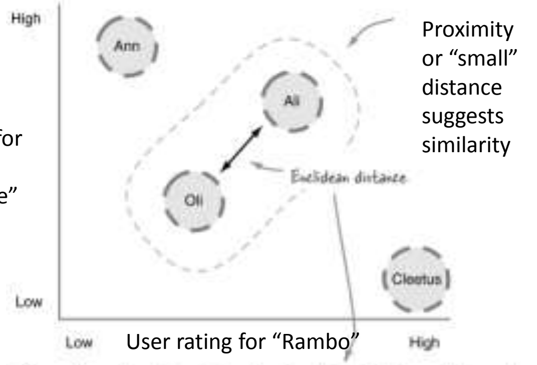
- ***Recommender systems***, which are also known as ***collaborative filtering (CF)*** **systems**, are the computer equivalent of you asking your friends for a restaurant recommendation.
- The more recommendations you receive from your friends, the higher the probability that you'll go there. In the online world you see recommender engines in play every day.
- There are two types of collaborative recommenders: user-based recommenders and item-based recommenders:
  - ***User-based recommenders*** look at users similar to a target user, and use their collaborative ratings to make predictions to the target user.
  - ***Item-based recommenders*** look at similar items, and use this information to recommend items that are related to items previously used by a target user.
- Both types of recommender systems need to be able to determine the **degree of similarity** between users or items, so we first need to look at how similarity metrics work.

Similarity Metrics

- In both user- and item-based recommenders, the system needs to find similar users or items. They do this by comparing users or items with each other to arrive at a similarity score.
- Popular measures that can calculate these scores include
  - Euclidean distance and
  - Pearson's correlation.
- These algorithms operate on numerical data, where the data points are vector-like (points in space).
- This means that information (data) must be transformed or represented in the numerical form before being processed.
- Mahout can support the Euclidean and Pearson's similarity measures for both user-based and item-based recommenders.
- Mahout supports additional similarity measures for item-based recommenders

Euclidean Distance

- The Euclidean distance is one of a family of related distance measures, which also includes the Manhattan distance (the distance between two points measured along axes at right angles)



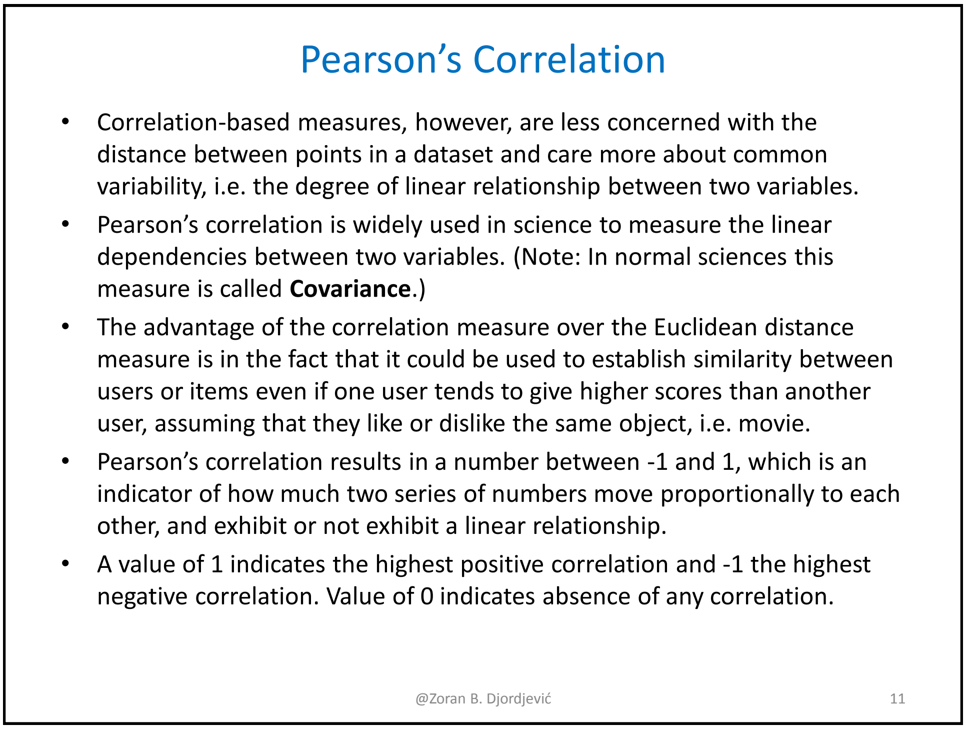
User rating for movie "Amelie"

Proximity or "small" distance suggests similarity

User rating for "Rambo"

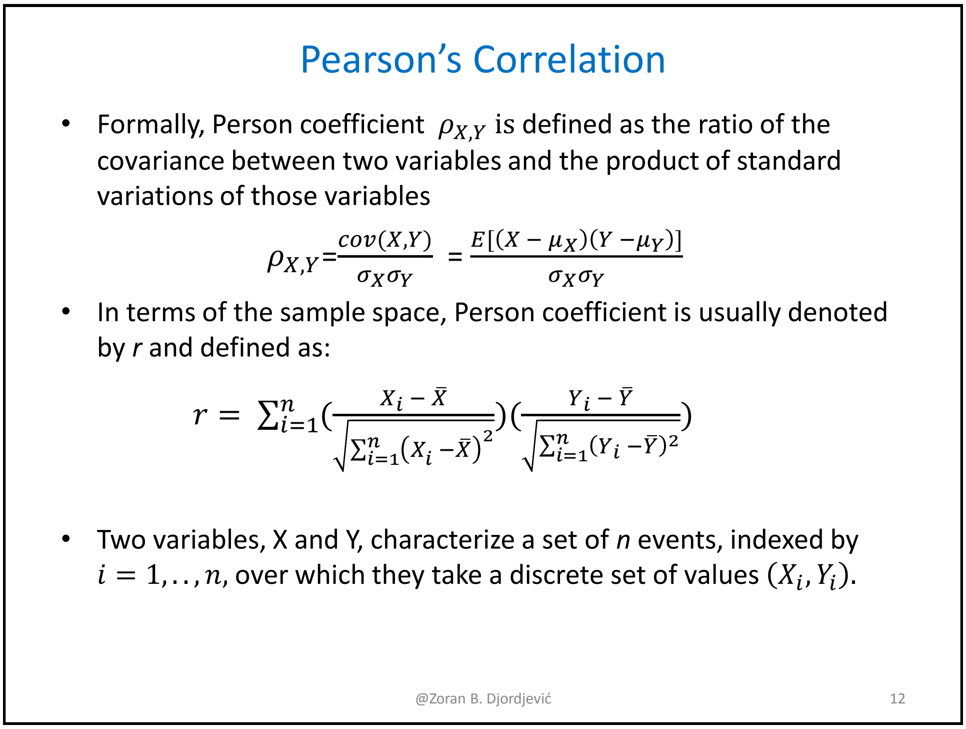
If $p = (p_1, p_2)$ and $q = (q_1, q_2)$, then the Euclidean distance is:

$$d(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2}$$



Pearson’s Correlation

- Correlation-based measures, however, are less concerned with the distance between points in a dataset and care more about common variability, i.e. the degree of linear relationship between two variables.
- Pearson’s correlation is widely used in science to measure the linear dependencies between two variables. (Note: In normal sciences this measure is called **Covariance**.)
- The advantage of the correlation measure over the Euclidean distance measure is in the fact that it could be used to establish similarity between users or items even if one user tends to give higher scores than another user, assuming that they like or dislike the same object, i.e. movie.
- Pearson’s correlation results in a number between -1 and 1, which is an indicator of how much two series of numbers move proportionally to each other, and exhibit or not exhibit a linear relationship.
- A value of 1 indicates the highest positive correlation and -1 the highest negative correlation. Value of 0 indicates absence of any correlation.



Pearson’s Correlation

- Formally, Person coefficient $\rho_{X,Y}$ is defined as the ratio of the covariance between two variables and the product of standard variations of those variables

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} = \frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y}$$

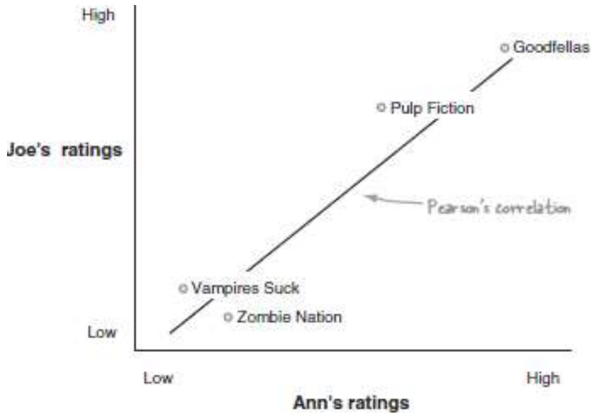
- In terms of the sample space, Person coefficient is usually denoted by *r* and defined as:

$$r = \sum_{i=1}^{n} \left(\frac{X_i - \bar{X}}{\sqrt{\sum_{i=1}^{n} (X_i - \bar{X})^2}} \right) \left(\frac{Y_i - \bar{Y}}{\sqrt{\sum_{i=1}^{n} (Y_i - \bar{Y})^2}} \right)$$

- Two variables, X and Y, characterize a set of *n* events, indexed by $i = 1, .., n$, over which they take a discrete set of values (X_i, Y_i).

Highly Correlated Relationship

- The diagram bellow suggest high level of correlation between scores given to movies by Joe and Ann. Y_i-s are scores (ratings) given by Joe and X_i-s scores (ratings) given by Ann.
- Highly elongated “cloud” of points suggests strong correlation and a Person coefficient which is close to 1.



Installing Mahout

- On the command prompt of your Hadoop VM, type:

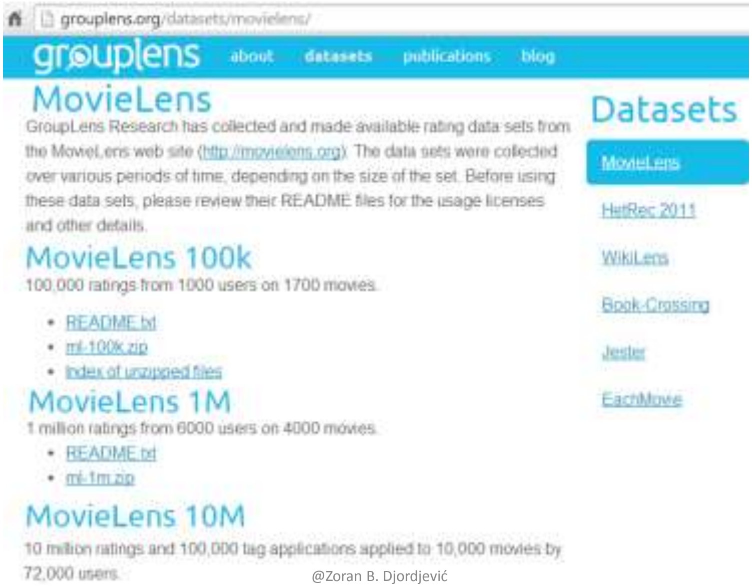
```
$ sudo yum install mahout
```

- If you get involved with Mahout, you will need to download newest libraries and rebuild code. This is done using Maven. Just in case, download and install Maven from http://maven.apache.org
- Mahout needs to know where Hadoop lives. Make sure you have `HADOOP_HOME` declared. Your `.bash_profile` file should look like:

```
JAVA_HOME=/usr/java/jdk1.7.0_51
export JAVA_HOME
MAVEN_HOME=/usr/lib/maven
export MAVEN_HOME
MAHOUT_HOME=/usr/lib/mahout
export MAHOUT_HOME
HADOOP_HOME=/usr/lib/hadoop
export HADOOP_HOME
PATH=$JAVA_HOME/bin:$PATH:$HOME/bin:$MAVEN_HOME/bin:$MAHOUT_HOME
export PATH
```

Recommending Movies, GroupLens dataset

http://www.grouplens.org/node/12 Started at Comp. Science Dept. of University of Minnesota



MovieLens

GroupLens Research has collected and made available rating data sets from the MovieLens web site (http://movielens.org). The data sets were collected over various periods of time, depending on the size of the set. Before using these data sets, please review their README files for the usage licenses and other details.

MovieLens 100k

100,000 ratings from 1000 users on 1700 movies.

- README.txt
- ml-100k.zip
- Index of unzipped files

MovieLens 1M

1 million ratings from 6000 users on 4000 movies.

- README.txt
- ml-1m.zip

MovieLens 10M

10 million ratings and 100,000 tag applications applied to 10,000 movies by 72,000 users.

Datasets

- MovieLens
- HetRec 2011
- WikiLens
- Book-Crossing
- Jester
- EachMovie

Download and prepare the data set

- GroupLens research lab provides a data set containing 10000054 ratings and 95580 tags applied to 10681 movies by 71567 users of the online movie recommender service MovieLens

```
$ cd $MAHOUT_HOME
$ mkdir -p corpus/grouplens-10m
$ cd corpus/grouplens-10m
```

- From `files.grouplens.org/datasets/movielens/ml-10m.zip` download 10 Million ratings file `ml-10m.zip`

```
$ sudo curl -O files.grouplens.org/datasets/movielens/ml-10m.zip
$ ls -la ml-10m.zip
-rw-r--r-- 1 root root 66296349 Apr 17 19:33 ml-10m.zip
$ sudo unzip ml-10m.zip
Archive: ml-10m.zip
creating: ml-10M100K/
inflating: ml-10M100K/allbut.pl
inflating: ml-10M100K/movies.dat
inflating: ml-10M100K/ratings.dat
inflating: ml-10M100K/README
inflating: ml-10M100K/split_ratings.sh
inflating: ml-10M100K/tags.dat
$cat ratings.dat | wc -l
10000054
```

`ratings.csv` file

- The ratings file contains data in the following format:

```
UserID::MovieID::Rating::Timestamp
```

- `UserIDs` range between 1 and 71567
- `MovieIDs` range between 1 and 10681
- `Ratings` are made on a 5-star scale (whole-star ratings only)
- `Timestamp` is represented in seconds since the epoch

- Each user made at least 20 ratings
- We can see an example from the top or bottom of the file:

```
$ head -n 5 ml-10M100K/ratings.dat   tail –n 5 ml-10m/rating.dat

    1::122::5::838985046          71567::2107::1::912580553
    1::185::5::838983525          71567::2126::2::912649143
    1::231::5::838983392          71567::2294::5::912577968
    1::292::5::838983421          71567::2338::2::912578016
    1::316::5::838983392          71567::2384::2::912578173
```

- Mahout expects CSV-delimited format like: `UserID,ItemID,Value`
- We can write an `awk` script to convert the GroupLens data into CSV format:

```
$ awk -F"::" '{print $1","$2","$3}' ml-10M100K/ratings.dat > ratings.csv
```

- The data is now in a form that Mahout could use:

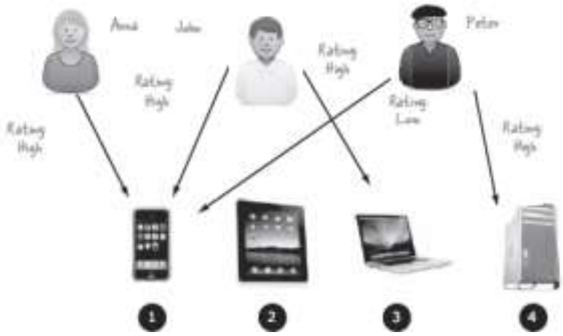
```
1,1193,5
1,661,3
1,914,3
```

Item-based Recommenders

- Mahout cannot currently run user-based recommenders in Map Reduce. Existing user-based recommender is designed to work within a single JVM.
- Item-based recommender can work with Map Reduce, though. This implies that you could run much bigger jobs for item-based analysis.
- Item-based recommenders look at similarities between item sets.
- You have recommended certain items with certain scores. Mahout locates other item groups which have similar items and similar item scores.
- Mahout recommends to you the items that people with similar items and score sets recommended (bought) and which you have not recommended (bought) yet. It could be that Amazon and other Internet vendors are doing just that.

Recommend items to Anna

- To recommend items to Anna, item recommendation looks at other items that co-occur with items Anna reviews (items 3 and 4), determines their similarity (based on reviewer ratings), and then ranks them by multiplying Anna's rating with the similarity rating for other items. In this example, item 3 would have a higher predicted value than item 4, because John rated 1 and 3 as high, but Peter rated 1 as low.



Make recommendations using Mahout

- Create a file `user-ids.txt` with ID-s of users to whom we want to make the recommendation which movie to watch next. Let the file have 3 users

```
$cat user-ids.txt
1
2
3
```

- Move files `user-ids.txt` and `ratings.csv` to the root of your HDFS directory

```
$ hadoop fs -put user-ids.txt .
$ hadoop fs -put ratings.csv .
```

- Run Mahout:

```
$ mahout recommenditembased -Dmapred.reduce.tasks=10 \
--similarityClassname SIMILARITY_PEARSON_CORRELATION \
--input ratings.csv --output item-rec-output \
--tempDir item-rec-tmp --usersFile user-ids.txt
```

Recommender’s Output

- If you examine your HDFS files, after the recommender’s job is finished, you will see 10 output files. One for each reducer we span.
- Only those with indexes 1, 2 and 3, corresponding to users 1, 2 and 3 have data in them.

```
cloudera@localhost ~]$ hadoop fs -ls item-rec-output
Found 12 items
-rw-r--r--   1 cloudera supergroup          0 2014-04-18 09:54 item-rec-output/_SUCCESS
drwxr-xr-x   - cloudera supergroup          0 2014-04-18 09:52 item-rec-output/_logs
-rw-r--r--   1 cloudera supergroup          0 2014-04-18 09:53 item-rec-output/part-r-00000
-rw-r--r--   1 cloudera supergroup         94 2014-04-18 09:53 item-rec-output/part-r-00001
-rw-r--r--   1 cloudera supergroup        125 2014-04-18 09:53 item-rec-output/part-r-00002
-rw-r--r--   1 cloudera supergroup         98 2014-04-18 09:53 item-rec-output/part-r-00003
-rw-r--r--   1 cloudera supergroup          0 2014-04-18 09:53 item-rec-output/part-r-00004
-rw-r--r--   1 cloudera supergroup          0 2014-04-18 09:53 item-rec-output/part-r-00005
-rw-r--r--   1 cloudera supergroup          0 2014-04-18 09:54 item-rec-output/part-r-00006
-rw-r--r--   1 cloudera supergroup          0 2014-04-18 09:54 item-rec-output/part-r-00007
-rw-r--r--   1 cloudera supergroup          0 2014-04-18 09:54 item-rec-output/part-r-00008
-rw-r--r--   1 cloudera supergroup          0 2014-04-18 09:54 item-rec-output/part-r-00009
[cloudera@localhost ~]$
```

Recommendations

- The output file consists of the user ID, followed by a comma-separated list of item IDs and their related scores:

```
$ hadoop fs -cat item-rec-output/part*
1 [1566:5.0,1036:5.0,1033:5.0,1032:5.0,1031:5.0,1030:5.0,3107:5.0,
3114:5.0,1026:5.0,1025:5.0]
2 [2739:5.0,3811:5.0,3916:5.0,2:5.0,10:5.0,11:5.0,16:5.0,3793:5.0,
3791:5.0,3789:5.0]
3 [1037:5.0,1036:5.0,2518:5.0,3175:5.0,3108:5.0,10:5.0,1028:5.0,
3104:5.0,1025:5.0,1019:5.0]
```

- You can verify that, unlike Amazon, Mahout does not recommend you watch movies you already watched or buy books you already bough.
- Item-based recommender creates a co-occurrence matrix to associate similar items together. It does this by combining items with similar ratings from each user, and then counting the number of times that each pair of items was rated by all the users.
- The recommender predicts the ratings for unknown items by multiplying the users’ ratings for an item with all the item’s co-occurrences, and then sorts all these item predictions and retains the top *K* as recommendations.

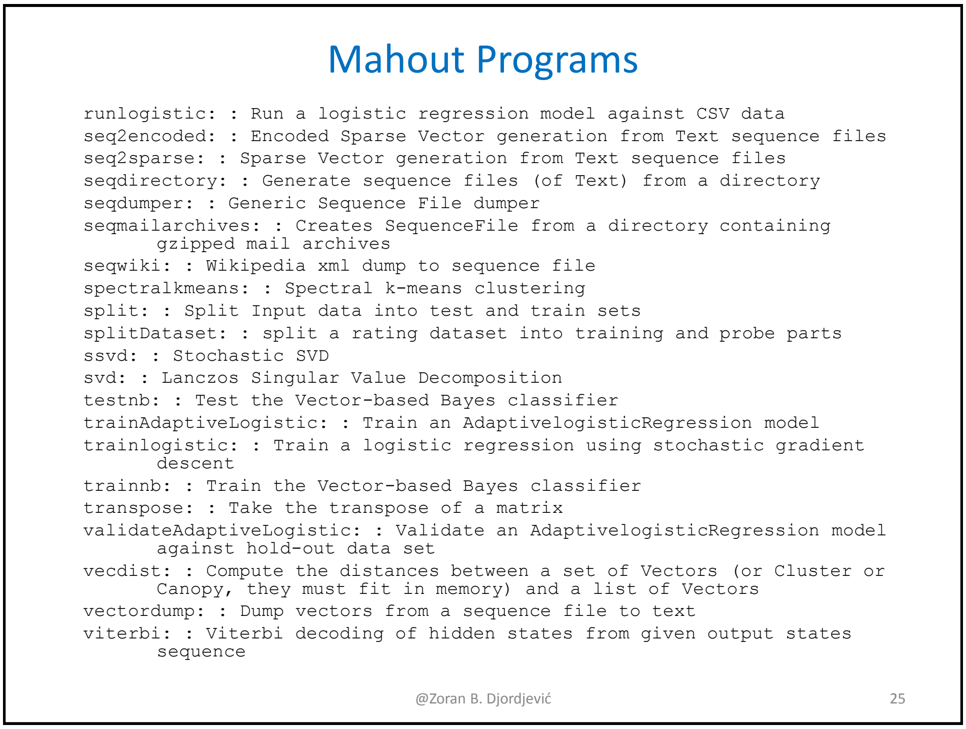
Mahout Programs

- Mahout has a number of built in functions or programs. You can list them all with brief descriptions if your type `$ mahout` on the command line:

```
[cloudera@localhost mahout]$ mahout
MAHOUT_LOCAL is not set; adding HADOOP_CONF_DIR to classpath.
Running on hadoop, using /usr/lib/hadoop/bin/hadoop and
HADOOP_CONF_DIR=/etc/hadoop/conf
MAHOUT-JOB: /usr/lib/mahout/mahout-examples-0.7-cdh4.6.0-job.jar
An example program must be given as the first argument.
Valid program names are:
  arff.vector: : Generate Vectors from an ARFF file or directory
  baumwelch: : Baum-Welch algorithm for unsupervised HMM training
  canopy: : Canopy clustering
  cat: : Print a file or resource as the logistic regression models would see it
  cleansvd: : Cleanup and verification of SVD output
  clusterdump: : Dump cluster output to text
  clusterpp: : Groups Clustering Output In Clusters
  cmdump: : Dump confusion matrix in HTML or text formats
  cvb: : LDA via Collapsed Variation Bayes (0th deriv. approx)
  cvb0_local: : LDA via Collapsed Variation Bayes, in memory locally.
  dirichlet: : Dirichlet Clustering
  eigencuts: : Eigencuts spectral clustering
```

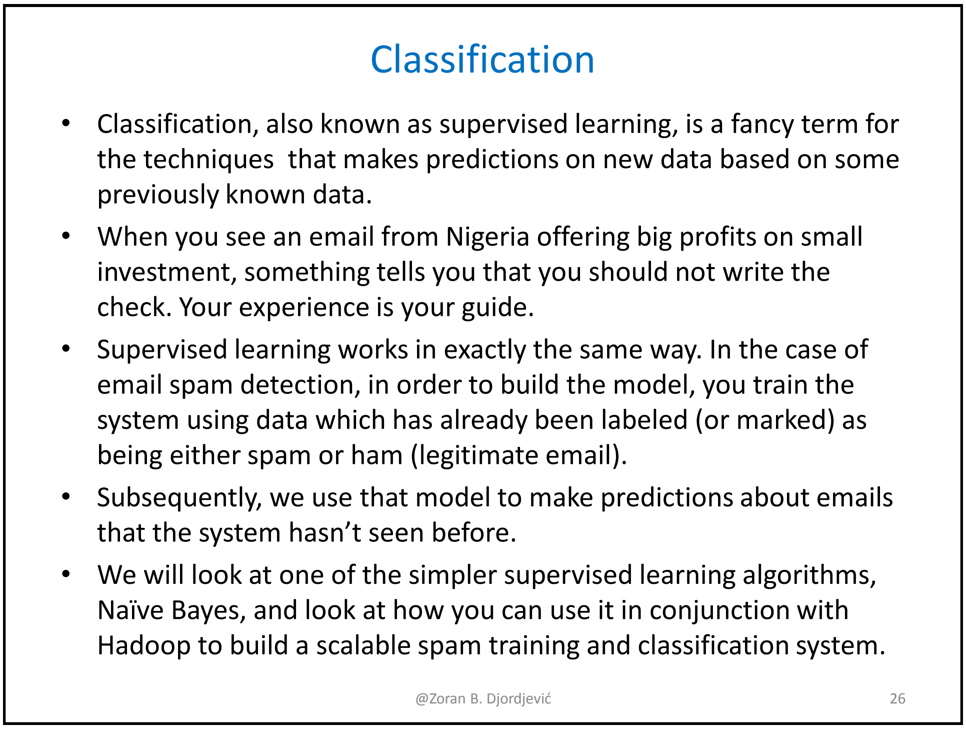
Mahout Programs

```
  evaluateFactorization: : compute RMSE and MAE of a rating matrix
         factorization against probes
  fkmeans: : Fuzzy K-means clustering
  fpg: : Frequent Pattern Growth
  hmmpredict: : Generate random sequence of observations by given HMM
  itemsimilarity: : Compute the item-item-similarities for item-based
                 collaborative filtering
  kmeans: : K-means clustering
  lucene.vector: : Generate Vectors from a Lucene index
  matrixdump: : Dump matrix in CSV format
  matrixmult: : Take the product of two matrices
  meanshift: : Mean Shift clustering
  minhash: : Run Minhash clustering
  parallelALS: : ALS-WR factorization of a rating matrix
  recommendfactorized: : Compute recommendations using the factorization of
         a rating matrix
  recommenditembased: : Compute recommendations using item-based
         collaborative filtering
  regexconverter: : Convert text files on a per line basis based on regular
         expressions
  rowid: : Map SequenceFile<Text,VectorWritable> to
{SequenceFile<IntWritable,VectorWritable>, SequenceFile<IntWritable,Text>}
  rowsimilarity: : Compute the pairwise similarities of the rows of a
         matrix
  runAdaptiveLogistic: : Score new production data using a probably trained
         and validated AdaptivelogisticRegression model
```



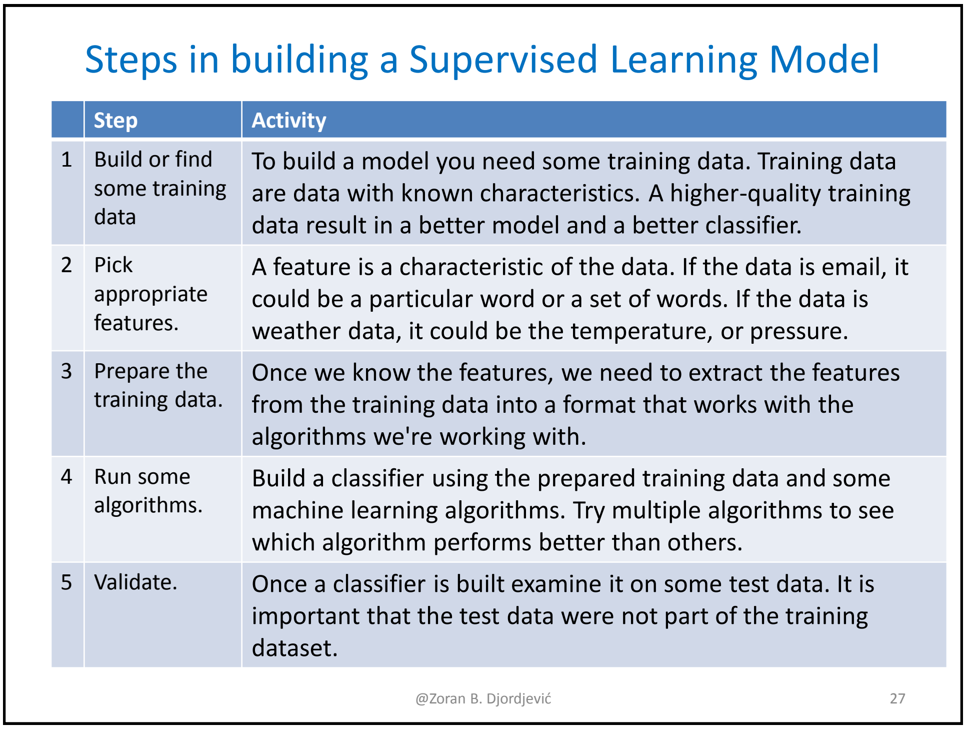
Mahout Programs

```
runlogistic: : Run a logistic regression model against CSV data
seq2encoded: : Encoded Sparse Vector generation from Text sequence files
seq2sparse: : Sparse Vector generation from Text sequence files
seqdirectory: : Generate sequence files (of Text) from a directory
seqdumper: : Generic Sequence File dumper
seqmailarchives: : Creates SequenceFile from a directory containing
       gzipped mail archives
seqwiki: : Wikipedia xml dump to sequence file
spectralkmeans: : Spectral k-means clustering
split: : Split Input data into test and train sets
splitDataset: : split a rating dataset into training and probe parts
ssvd: : Stochastic SVD
svd: : Lanczos Singular Value Decomposition
testnb: : Test the Vector-based Bayes classifier
trainAdaptiveLogistic: : Train an AdaptivelogisticRegression model
trainlogistic: : Train a logistic regression using stochastic gradient
       descent
trainnb: : Train the Vector-based Bayes classifier
transpose: : Take the transpose of a matrix
validateAdaptiveLogistic: : Validate an AdaptivelogisticRegression model
       against hold-out data set
vecdist: : Compute the distances between a set of Vectors (or Cluster or
       Canopy, they must fit in memory) and a list of Vectors
vectordump: : Dump vectors from a sequence file to text
viterbi: : Viterbi decoding of hidden states from given output states
       sequence
```



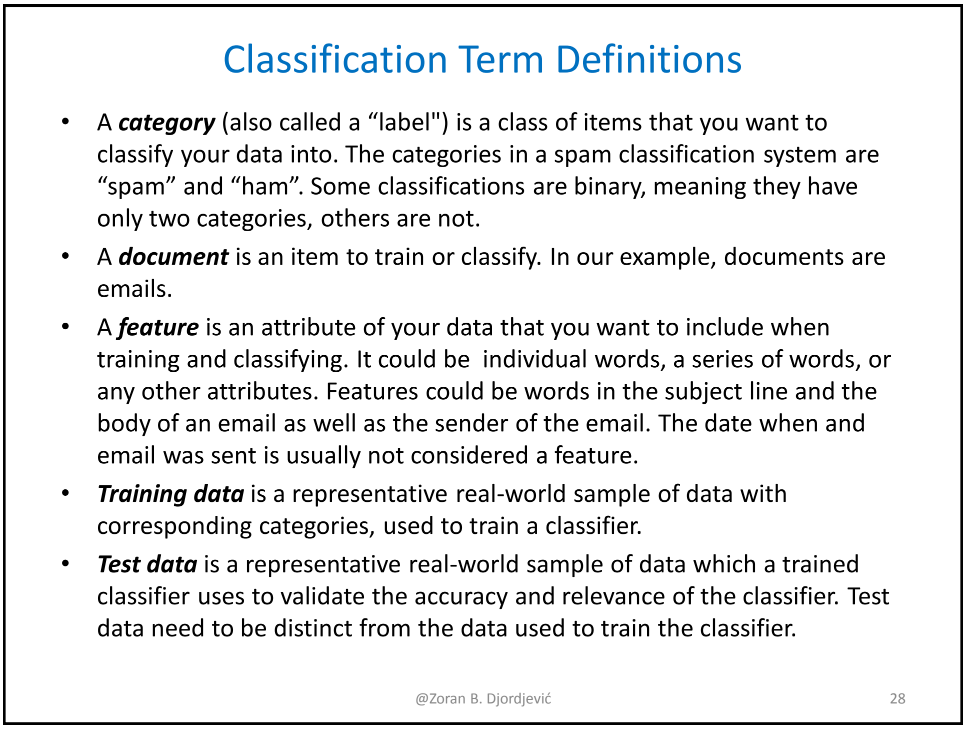
Classification

- Classification, also known as supervised learning, is a fancy term for the techniques that makes predictions on new data based on some previously known data.
- When you see an email from Nigeria offering big profits on small investment, something tells you that you should not write the check. Your experience is your guide.
- Supervised learning works in exactly the same way. In the case of email spam detection, in order to build the model, you train the system using data which has already been labeled (or marked) as being either spam or ham (legitimate email).
- Subsequently, we use that model to make predictions about emails that the system hasn’t seen before.
- We will look at one of the simpler supervised learning algorithms, Naïve Bayes, and look at how you can use it in conjunction with Hadoop to build a scalable spam training and classification system.



Steps in building a Supervised Learning Model

	Step	Activity
1	Build or find some training data	To build a model you need some training data. Training data are data with known characteristics. A higher-quality training data result in a better model and a better classifier.
2	Pick appropriate features.	A feature is a characteristic of the data. If the data is email, it could be a particular word or a set of words. If the data is weather data, it could be the temperature, or pressure.
3	Prepare the training data.	Once we know the features, we need to extract the features from the training data into a format that works with the algorithms we're working with.
4	Run some algorithms.	Build a classifier using the prepared training data and some machine learning algorithms. Try multiple algorithms to see which algorithm performs better than others.
5	Validate.	Once a classifier is built examine it on some test data. It is important that the test data were not part of the training dataset.



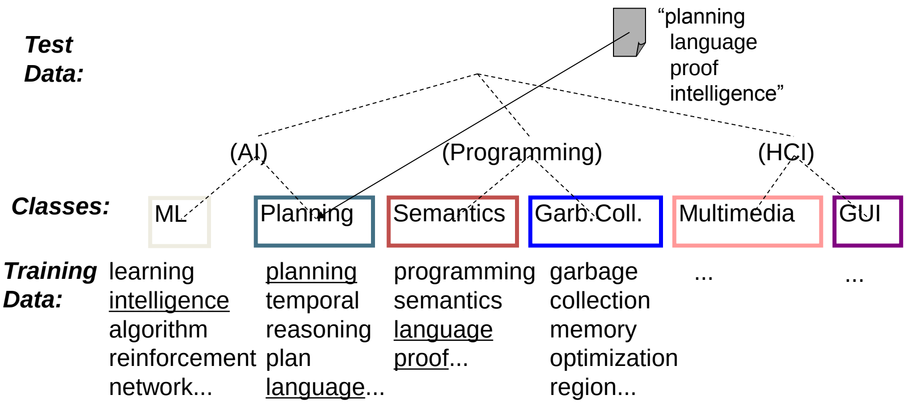
Classification Term Definitions

- A ***category*** (also called a “label") is a class of items that you want to classify your data into. The categories in a spam classification system are “spam” and “ham”. Some classifications are binary, meaning they have only two categories, others are not.
- A ***document*** is an item to train or classify. In our example, documents are emails.
- A ***feature*** is an attribute of your data that you want to include when training and classifying. It could be individual words, a series of words, or any other attributes. Features could be words in the subject line and the body of an email as well as the sender of the email. The date when and email was sent is usually not considered a feature.
- ***Training data*** is a representative real-world sample of data with corresponding categories, used to train a classifier.
- ***Test data*** is a representative real-world sample of data which a trained classifier uses to validate the accuracy and relevance of the classifier. Test data need to be distinct from the data used to train the classifier.

More formally: Supervised Classification

- When statisticians or even serious computer scientists describe classification the language could be quite dry. For example, they would describe supervised learning in this manner:
- Given:
  - A description of an instance, $d \epsilon X$
    - X is the instance language or instance space.
  - A fixed set of classes:
    $C \ = \{c_1, c_2, \dots, c_J\}$
  - A training set D of labeled documents with each labeled document $\langle d, c \rangle \ \in \ X \times C$
- Determine:
  - A learning method or algorithm which will enable us to learn a classifier $\gamma: X \rightarrow C$
  - For a test document d, we assign it the class $\gamma(d) \ \in \ C$

Classification of Documents on Computer Science Topics



Test Data:

"planning language proof intelligence"

(AI) (Programming) (HCI)

Classes: ML | Planning | Semantics | Garb.Coll. | Multimedia | GUI

Training Data:

ML: learning <u>intelligence</u> algorithm reinforcement network...

Planning: <u>planning</u> temporal reasoning plan <u>language</u>...

Semantics: programming semantics <u>language</u> <u>proof</u>...

Garb.Coll.: garbage collection memory optimization region...

Multimedia: ...

GUI: ...

- Classification is not always flat.
- In real life, items (documents) are often naturally organized in hierarchies. Many classifications algorithms could extract those hierarchies.

Categories are Labels

- Labels are most often topics such as Yahoo-categories
  - "finance," "sports," "news>world>asia>business"
- Labels may be genres
  - "editorials" "movie-reviews" "news”
- Labels may be opinions on a person/product
  - “like”, “hate”, “neutral”
- Labels may be domain-specific
  - "interesting-to-me" : "not-interesting-to-me”
  - “contain medical information” : “doesn’t”
  - language identification: English, French, Chinese, …
  - search vertical: “about Linux”, “about Windows”, ”SunOS”,…
  - “spam” : “not spam”

Ch. 13

Classification Methods

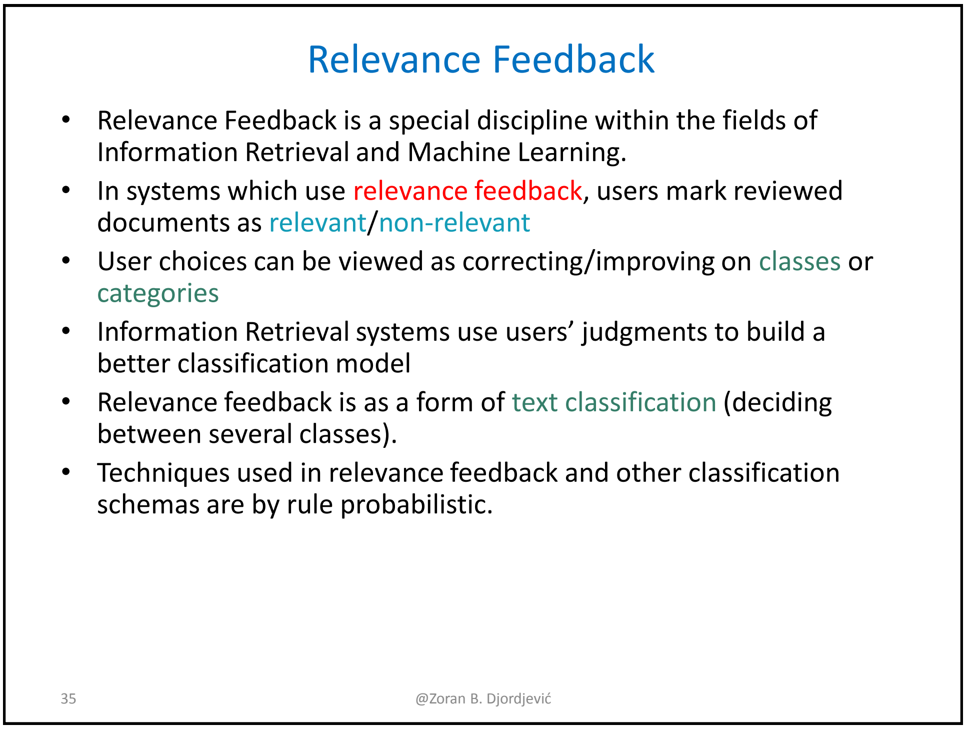
- Manual classification
  - Used by the original Yahoo! Directory
  - Looksmart, About.com, ODP, PubMed
  - Could be very accurate when job is done by experts
  - Consistent when the problem size and the team are small
  - Difficult and expensive to scale. Yahoo did try manual classification of the Web originally, before Google came up with Page-Rang technology
    - Means we need automatic classification methods for big problems

Classification Methods (2)

- Hand-coded rule-based classifiers
  - Technique used by simple spam filters, Reuters, CIA, etc.
  - Widely used in government, **legal firms** and commercial enterprises
  - Companies used to build “IDE”-s for writing such rules
  - A rule-based classifier assigns a document to a category if document contains a given Boolean combination of words
  - Commercial systems have complex query languages (at the time complex Information Retrieval query languages + score accumulators were developed)
  - Accuracy is often very high if a rule has been carefully refined over time by a subject expert
  - Building and maintaining these rules is expensive

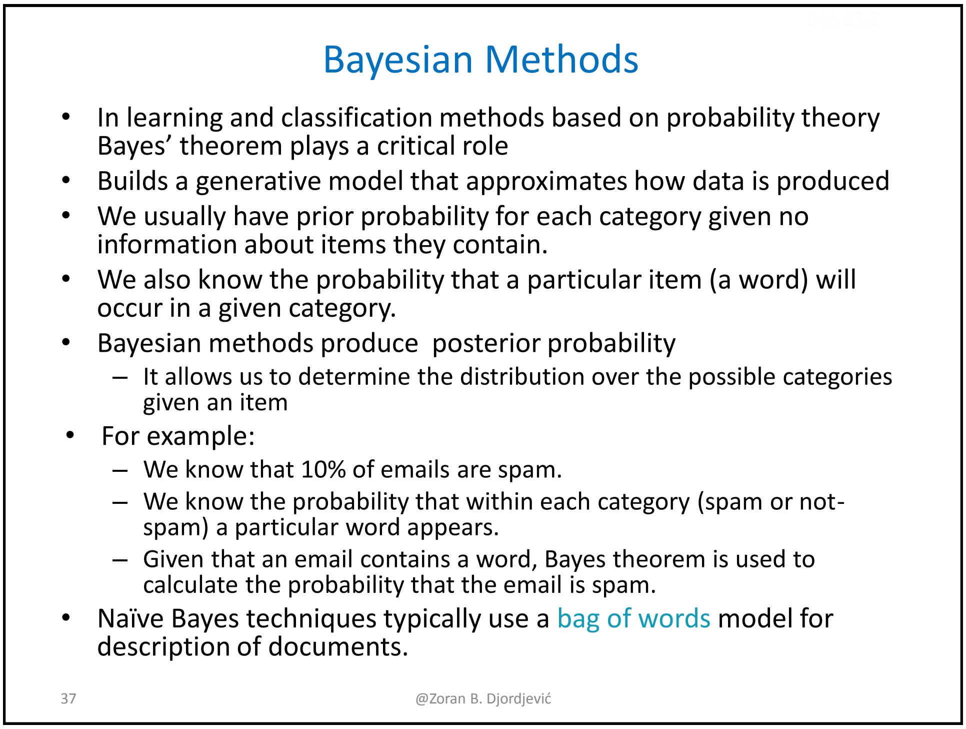
Classification Methods (3)

- Supervised learning was used for the document-label assignment function
  - Many systems partly or wholly rely on machine learning (Autonomy, Microsoft, Enkata, Yahoo!, …)
    - k-Nearest Neighbors (simple, powerful)
    - Naive Bayes (simple, common method)
    - Support-vector machines (new, generally more powerful)
    - … plus many other methods
  - No free lunch: requires hand-classified training data
  - But data can be built up (and refined) by amateurs
- Many commercial systems use a mixture of methods



Relevance Feedback

- Relevance Feedback is a special discipline within the fields of Information Retrieval and Machine Learning.
- In systems which use relevance feedback, users mark reviewed documents as relevant/non-relevant
- User choices can be viewed as correcting/improving on classes or categories
- Information Retrieval systems use users' judgments to build a better classification model
- Relevance feedback is as a form of text classification (deciding between several classes).
- Techniques used in relevance feedback and other classification schemas are by rule probabilistic.



Sec.13.2

Bayesian Methods

- In learning and classification methods based on probability theory Bayes' theorem plays a critical role
- Builds a generative model that approximates how data is produced
- We usually have prior probability for each category given no information about items they contain.
- We also know the probability that a particular item (a word) will occur in a given category.
- Bayesian methods produce posterior probability
  - It allows us to determine the distribution over the possible categories given an item
- For example:
  - We know that 10% of emails are spam.
  - We know the probability that within each category (spam or not-spam) a particular word appears.
  - Given that an email contains a word, Bayes theorem is used to calculate the probability that the email is spam.
- Naïve Bayes techniques typically use a bag of words model for description of documents.

Example of use of Bayes Theorem

- Public officials estimate the incidence of the HIV virus in the general population is about 0.5 percent.
- A test is introduced which when given to people with HIV correctly identifies the virus 95 percent of the time.
- The test also gives a false positive 5 percent of the time. The test result could be positive even though the person does not have HIV.
- Of the people who test positive, what percent of them we actually expect to have the virus.

- A SIDE NOTE: sometime we use the language of odds. Odds are the ratios of number of incidents of a positive event and the number of incidents of a negative event. In terms of probabilities odds can be expressed as:

$$O(a) = \frac{p(a)}{p(\bar{a})} = \frac{p(a)}{1 - p(a)}$$

Bayes Solution

- Probability that a person has HIV: $P(HIV) = 0.005$
- Given HIV in a patient, test identifies the virus 95% of the time $P(POS|HIV) = 0.95$
- Given that a person does not have HIV, test will make a false positive claim 5% of the time: $P(POS|\overline{HIV}) = 0.05$
- Given that test gave a positive result, the probability that the patient has HIV is:

$$P(HIV|POS) = \frac{P(POS|HIV)P(HIV)}{P(POS)}$$
$$= \frac{P(POS|HIV)P(HIV)}{P(POS|HIV)P(HIV) + \ P(POS|\overline{HIV})P(\overline{HIV})}$$
$$= \frac{0.95 * 0.005}{0.95 * 0.005 + 0.05 * (1 - 0.005)} = \frac{0.00475}{0.00475 + 0.04975}$$
$$\approx 0.0872$$

- Approximately **8.72%** of people who test positive will have the virus.

Classification Function $\gamma(d)$acting on the Document

- Classification function determines the class to which a document belongs

γ(

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun… It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet.

)=c

Document is treated as a bag of words

- Typically, the order of words is considered non-relevant.
- We just count how many times a word appears in a document.
- Classification functions similarly do not depend on anything but word counts

γ(

great	2
love	2
recommend	1
laugh	1
happy	1
. . .	. . .

)=c

Sec.13.2

Bayes’ Rule for text classification

- For a document d and a class c, using Bayes theorem, we could state that probability $P(c|d)$, i.e. the probability that category c happens, given document d, can be expressed as

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)}$$

- where $P(d|c)$ *is* the probability that document d belongs to the category c, (i.e. probability that d happens given c) and prior probability $P(d)$ that document d happens, and the prior probability $P(c)$ that category c happens.
- $P(d)$ is simply $1/N_d$, where N_d is the number of documents.
- $P(c)$ is similarly $1/N_c$, where N_c is the number of classes.
- In order to find $P(c|d)$, the job is reduced to determining $P(d|c)$.
- Typically, the denominator $P(d)$ is ignored.

Sec.13.2

Naive Bayes Classifiers

Documents are represented as collections (bags) of words

Task: Classify a new instance d based on a tuple of attribute values $d(x_1, x_2, .., x_n)$ into one of the classes $c_j \in C$

The goal is to find the best class for the document

$$c_{MAP} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j \mid x_1, x_2, K\ , x_n)$$

$$= \underset{c_j \in C}{\operatorname{argmax}} \frac{P(x_1, x_2, K\ , x_n \mid c_j) P(c_j)}{P(x_1, x_2, K\ , x_n)}$$

$$= \underset{c_j \in C}{\operatorname{argmax}} P(x_1, x_2, K\ , x_n \mid c_j) P(c_j)$$

MAP is “maximum a posteriori” = most likely class

Sec.13.2

Naïve Bayes Classifier: Naïve Bayes Assumption

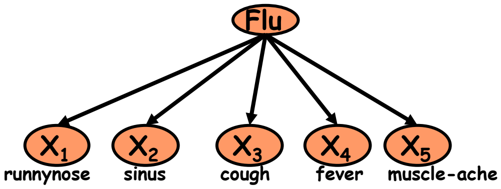
- $P(c_j)$ Can be estimated from the frequency of classes in the training examples.
- $P(x_1,x_2,\ldots,x_n|c_j)$
  - $O(|X|^n \bullet |C|)$ parameters
  - Could only be estimated if a very, very large number of training examples was available.

Naïve Bayes Conditional Independence Assumption:

- Assume that the probability of observing the conjunction of attributes is equal to the product of the individual probabilities $P(x_i|c_j)$.

Sec.13.3

The Multivariate Bernoulli NB Classifier



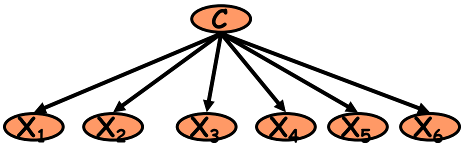
- Consider you are given a document with 5 words (terms) and you want to find the probability the that document is in the Flue class of documents.
- **Conditional Independence Assumption:** feature (term) presences are independent of each other given the class:

$$P(x_1,x_2,..,x_5|\ C) = P(x_1|C)\ P(x_2|C)\ldots\ P(x_5|C)$$

- We are saying if a text contains words: $x1$, $x2$, .., $x5$, we can estimate its probability to be in the class C as the products of probabilities that each word in the document is found in documents known to belong to class C.
- Probabilities on the right are easy to calculate. We need to look at all documents in the training set and find probabilities that every particular word is found among the words of that training set.

Sec.13.3

Learning the Model



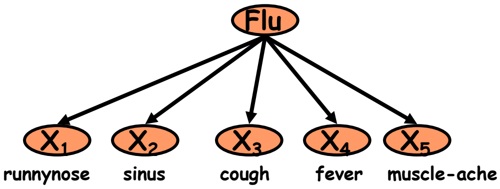
- First attempt: Simply use the frequencies in the data
- Probability that word $c_j$occurs is the ratio of the number of occurrences of the word $N(c = c_j)$, divided by the number of documents N.

$$\hat{P}(c_j) = \frac{N(C = c_j)}{N}$$

$$\hat{P}(x_i \mid c_j) = \frac{N(X_i = x_i, C = c_j)}{N(C = c_j)}$$

Sec.13.3

Problem with Simple Approach



- What if we have seen no training documents with the word **muscle-ache** classified in the category **Flu**?

$$P(x_1, x_2, .., x_5 \mid C) = P(x_1|C)\ P(x_2|C) \ldots P(x_5|C)$$

$$\hat{P}(X_5 = t \mid C = nf) = \frac{N(X_5 = t, C = nf)}{N(C = nf)} = 0$$

- Zero probabilities cannot be conditioned away, no matter the other evidence! The fact that the last probability is zero will eliminate (set to zero) the probability that the above document belongs to the Flue category.

Sec.13.3

Smoothing to Avoid Over-fitting

- One way to deal with the issue is to pretend that every word is present at least once in every class.

$$\hat{P}(x_i \mid c_j) = \frac{N(X_i = x_i, C = c_j) + 1}{N(C = c_j) + k}$$

# of values of X_i

- There are more elaborate schemas for improving the estimate of both $P(x_i|c_j)$ and $P(x_1, \ldots, x_n|C_j)$.
- For example, rather than estimating $P(x_1, \ldots, x_n|C_j)$ only on the basis of probabilities for appearance of single words, we can include the probabilities for occurrence of word pairs, bigrams, sequences of three word, trigrams and so forth.
- Mahout implements both simple and more complex techniques.

Sec. 9.1.2

Probabilistic relevance feedback

- Similar probabilistic techniques are used in assessing relevance feedback.
- If user has told us that some documents are relevant and some are non-relevant, then we can proceed to build a probabilistic classifier
  - such as the Naive Bayes model:
    - $P(t_k|R) = |\boldsymbol{D}_{rk}| / |\boldsymbol{D}_r|$
    - $P(t_k|NR) = |\boldsymbol{D}_{nrk}| / |\boldsymbol{D}_{nr}|$
  - t_k is a term;
  - $\boldsymbol{D}_r$ is the set of known relevant documents;
  - $\boldsymbol{D}_{rk}$ is the subset that contain t_k;
  - $\boldsymbol{D}_{nr}$ is the set of known non-relevant documents;
  - $\boldsymbol{D}_{nrk}$ is the subset that contain t_k.

Using Mahout to Train and Test a Spam Classifier

- With Mahout we can train a spam classifier using large volumes of emails and test how well that classifier performs on unseen emails?

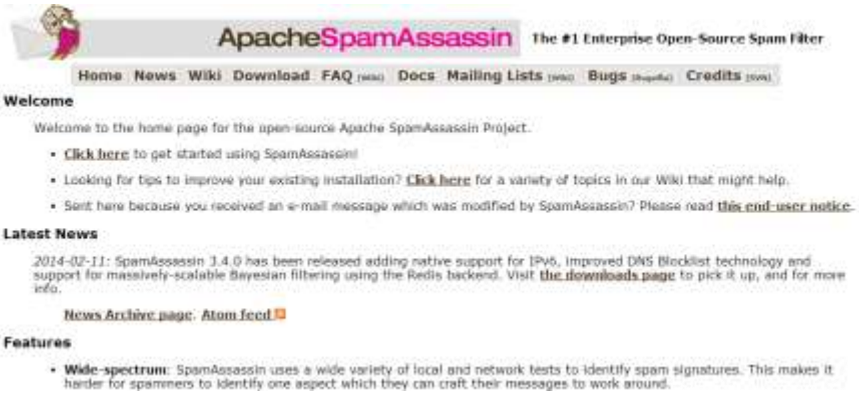
Problem

- You have a large spam corpus and you want to build a spam classifier.

Solution

- Use Mahout's naïve Bayes classifier on the Spam Assassin corpus to train a model, and then test its effectiveness on unseen emails.
- The SpamAssassin corpus, can be downloaded from http://spamassassin.apache.org/publiccorpus/
- We use this corpus both to train our classifier and to test it to see how well it performs at detecting spam.
- The first step is to download and extract the spam and ham datasets:

Apache Spam Assassin



ApacheSpamAssassin
The #1 Enterprise Open-Source Spam Filter
Home News Wiki Download FAQ (wiki) Docs Mailing Lists (wiki) Bugs (Bugzilla) Credits (SVN)

Welcome

Welcome to the home page for the open-source Apache SpamAssassin Project.

- Click here to get started using SpamAssassin!
- Looking for tips to improve your existing installation? Click here for a variety of topics in our Wiki that might help.
- Sent here because you received an e-mail message which was modified by SpamAssassin? Please read this end-user notice.

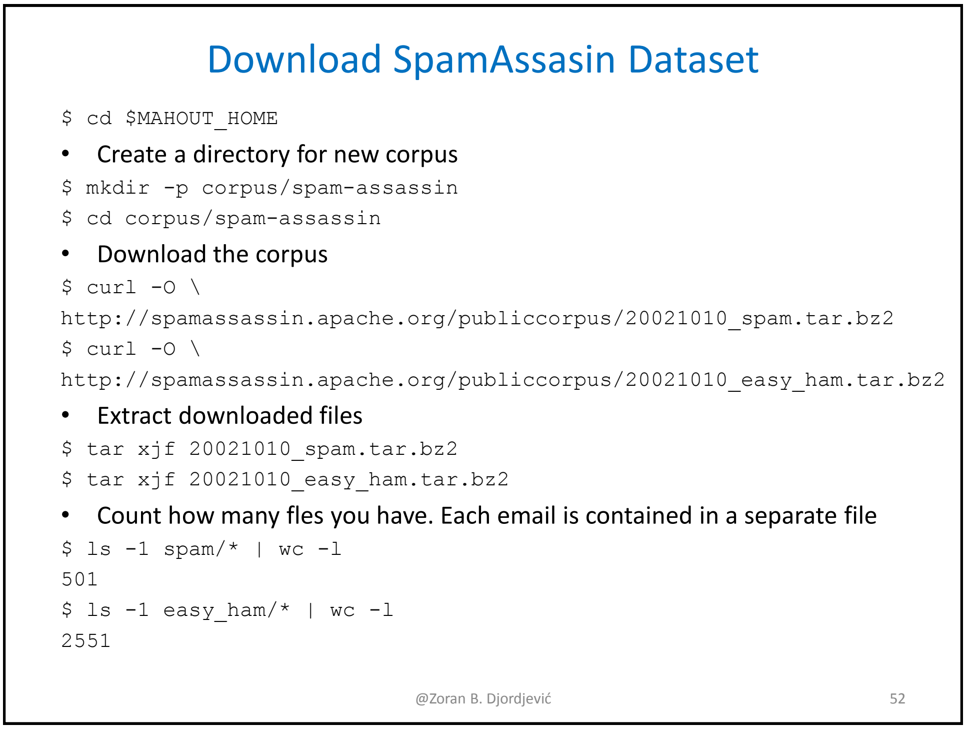
Latest News

2014-02-11: SpamAssassin 3.4.0 has been released adding native support for IPv6, improved DNS Blocklist technology and support for massively-scalable Bayesian filtering using the Redis backend. Visit the downloads page to pick it up, and for more info.

News Archive page, Atom feed

Features

- **Wide-spectrum**: SpamAssassin uses a wide variety of local and network tests to identify spam signatures. This makes it harder for spammers to identify one aspect which they can craft their messages to work around.



Download SpamAssasin Dataset

```
$ cd $MAHOUT_HOME
```

- Create a directory for new corpus

```
$ mkdir -p corpus/spam-assassin
$ cd corpus/spam-assassin
```

- Download the corpus

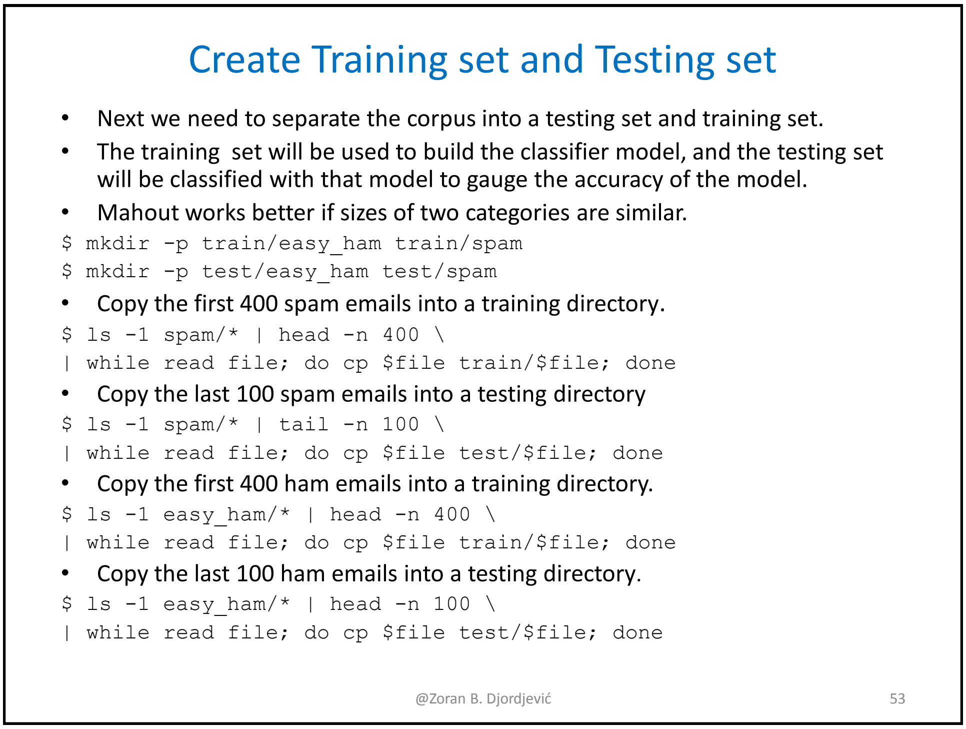
```
$ curl -O \
http://spamassassin.apache.org/publiccorpus/20021010_spam.tar.bz2
$ curl -O \
http://spamassassin.apache.org/publiccorpus/20021010_easy_ham.tar.bz2
```

- Extract downloaded files

```
$ tar xjf 20021010_spam.tar.bz2
$ tar xjf 20021010_easy_ham.tar.bz2
```

- Count how many fles you have. Each email is contained in a separate file

```
$ ls -1 spam/* | wc -l
501
$ ls -1 easy_ham/* | wc -l
2551
```



Create Training set and Testing set

- Next we need to separate the corpus into a testing set and training set.
- The training set will be used to build the classifier model, and the testing set will be classified with that model to gauge the accuracy of the model.
- Mahout works better if sizes of two categories are similar.

```
$ mkdir -p train/easy_ham train/spam
$ mkdir -p test/easy_ham test/spam
```

- Copy the first 400 spam emails into a training directory.

```
$ ls -1 spam/* | head -n 400 \
| while read file; do cp $file train/$file; done
```

- Copy the last 100 spam emails into a testing directory

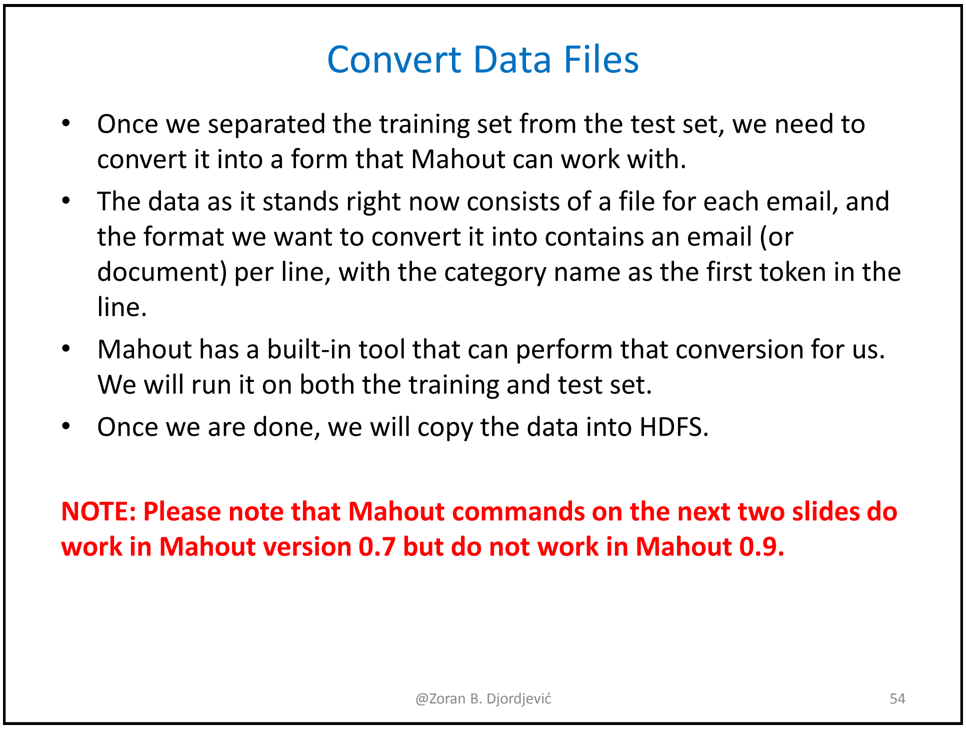
```
$ ls -1 spam/* | tail -n 100 \
| while read file; do cp $file test/$file; done
```

- Copy the first 400 ham emails into a training directory.

```
$ ls -1 easy_ham/* | head -n 400 \
| while read file; do cp $file train/$file; done
```

- Copy the last 100 ham emails into a testing directory.

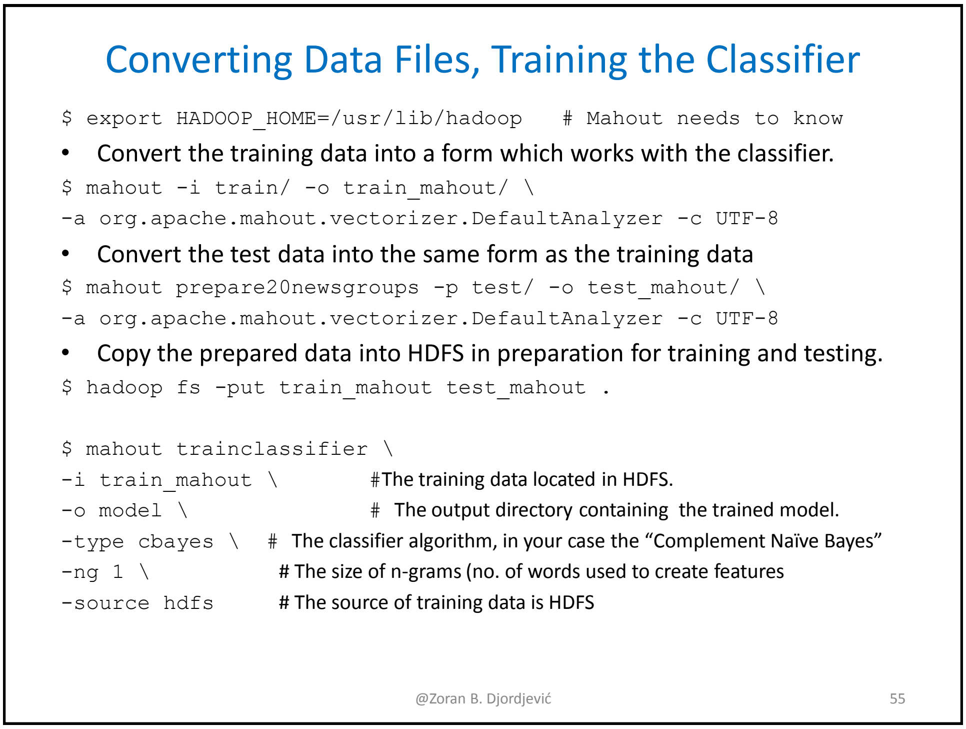
```
$ ls -1 easy_ham/* | head -n 100 \
| while read file; do cp $file test/$file; done
```



Convert Data Files

- Once we separated the training set from the test set, we need to convert it into a form that Mahout can work with.
- The data as it stands right now consists of a file for each email, and the format we want to convert it into contains an email (or document) per line, with the category name as the first token in the line.
- Mahout has a built-in tool that can perform that conversion for us. We will run it on both the training and test set.
- Once we are done, we will copy the data into HDFS.

NOTE: Please note that Mahout commands on the next two slides do work in Mahout version 0.7 but do not work in Mahout 0.9.



Converting Data Files, Training the Classifier

```
$ export HADOOP_HOME=/usr/lib/hadoop   # Mahout needs to know
```

- Convert the training data into a form which works with the classifier.

```
$ mahout -i train/ -o train_mahout/ \
-a org.apache.mahout.vectorizer.DefaultAnalyzer -c UTF-8
```

- Convert the test data into the same form as the training data

```
$ mahout prepare20newsgroups -p test/ -o test_mahout/ \
-a org.apache.mahout.vectorizer.DefaultAnalyzer -c UTF-8
```

- Copy the prepared data into HDFS in preparation for training and testing.

```
$ hadoop fs -put train_mahout test_mahout .

$ mahout trainclassifier \
-i train_mahout \         #The training data located in HDFS.
-o model \                #  The output directory containing  the trained model.
-type cbayes \   #  The classifier algorithm, in your case the “Complement Naïve Bayes”
-ng 1 \          # The size of n-grams (no. of words used to create features
-source hdfs     # The source of training data is HDFS
```

Testing the Classifier

- When the training has completed it persists the model into several subdirectories under the model directory in HDFS.
- We can now run the classifier on the test data by specifying the location of our model

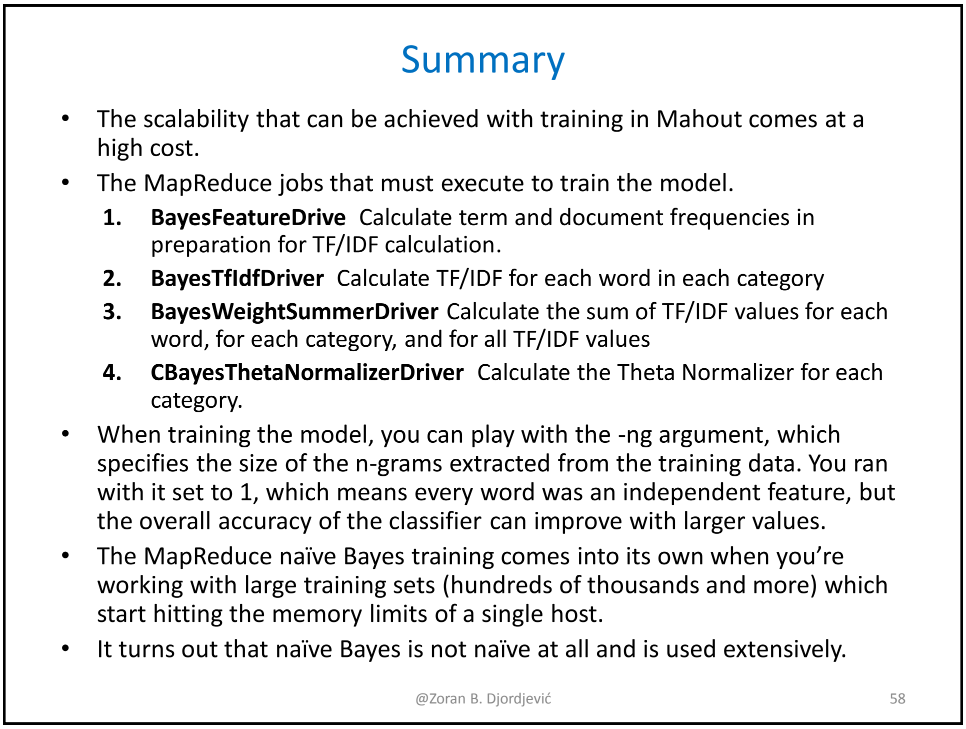
```
$ mahout testclassifier
-d test_mahout      #  The HDFS directory containing the test data.
-m model \          #  The HDFS directory containing the trained model.
-type cbayes \     #  The classifier algorithm, in our case the “Complement Naïve Bayes”
-ng 1 \             #  the size of the n-grams used
-source hdfs \     #  the source of the training data, HDFS
-method mapreduce  #  runtime system, alternative is “sequential” in a single JVM
```

Confusion Matrix

- The output of the *testclassifier* command shows something called a *confusion matrix*, which is telling you that the classifier correctly identified 72 spam emails as being spam, and incorrectly identified 28 spam emails as ham.

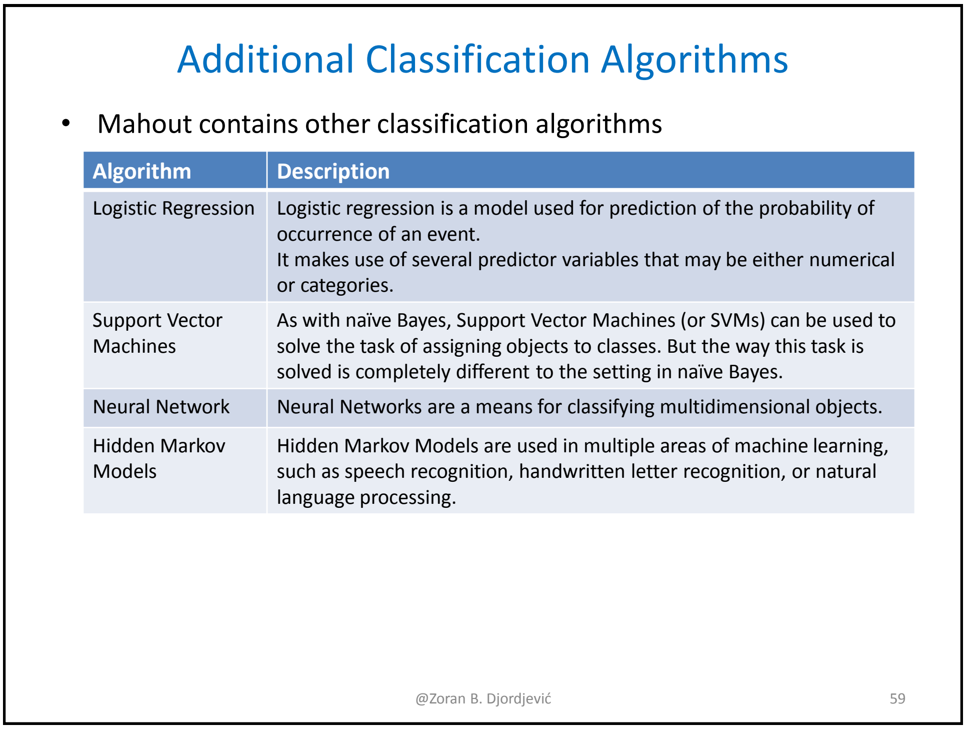
```
Confusion Matrix
-------------------------------------------------------
a      b   <--Classified as
72     28  | 100 a = spam
0      100 | 100 b = easy_ham
```

- Your classifier also was able to be 100 percent accurate at classifying ham emails.



Summary

- The scalability that can be achieved with training in Mahout comes at a high cost.
- The MapReduce jobs that must execute to train the model.
  1. **BayesFeatureDrive** Calculate term and document frequencies in preparation for TF/IDF calculation.
  2. **BayesTfIdfDriver** Calculate TF/IDF for each word in each category
  3. **BayesWeightSummerDriver** Calculate the sum of TF/IDF values for each word, for each category, and for all TF/IDF values
  4. **CBayesThetaNormalizerDriver** Calculate the Theta Normalizer for each category.
- When training the model, you can play with the -ng argument, which specifies the size of the n-grams extracted from the training data. You ran with it set to 1, which means every word was an independent feature, but the overall accuracy of the classifier can improve with larger values.
- The MapReduce naïve Bayes training comes into its own when you're working with large training sets (hundreds of thousands and more) which start hitting the memory limits of a single host.
- It turns out that naïve Bayes is not naïve at all and is used extensively.



Additional Classification Algorithms

- Mahout contains other classification algorithms

Algorithm	Description
Logistic Regression	Logistic regression is a model used for prediction of the probability of occurrence of an event. It makes use of several predictor variables that may be either numerical or categories.
Support Vector Machines	As with naïve Bayes, Support Vector Machines (or SVMs) can be used to solve the task of assigning objects to classes. But the way this task is solved is completely different to the setting in naïve Bayes.
Neural Network	Neural Networks are a means for classifying multidimensional objects.
Hidden Markov Models	Hidden Markov Models are used in multiple areas of machine learning, such as speech recognition, handwritten letter recognition, or natural language processing.

Sec. 6.2.2

tf-idf weighting

- The tf-idf weight of a term is the product of its tf weight and its idf weight.

$$w_{t,d} = \log(1+\mathrm{tf}_{t,d}) \times \log_{10}(N/\mathrm{df}_t)$$

- Best known weighting scheme in information retrieval
  - Note: the “-” in tf-idf is a hyphen, not a minus sign!
  - Alternative names: tf.idf, tf x idf
- Increases with the number of occurrences within a document
- Increases with the rarity of the term in the collection

Clustering with K-means

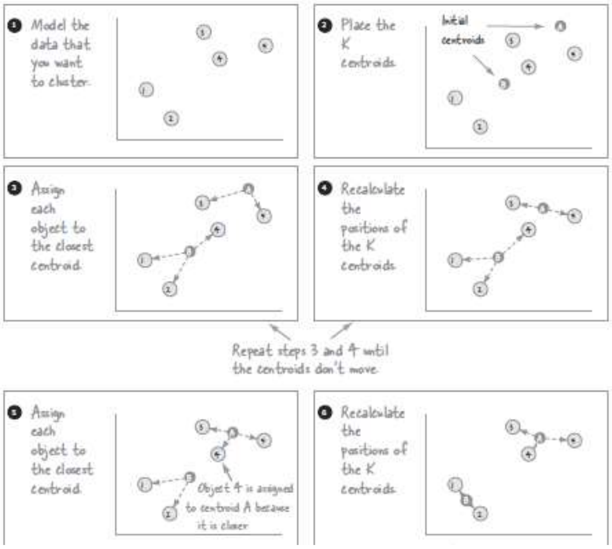
- Why Clustering?
- Quite often we do not human created classification categories but still want to examine whether our dataset separates in groups, i.e. clusters.
- Mechanically identified clusters often serve as the basis for human refinement. Sometimes they are used as is as the foundation of classification algorithms.
- K-means is one of the basic clustering algorithms.
- As with many algorithms in Mahout, K-means also has both sequential (in-memory) and parallel (MapReduce) implementations.
- To demonstrate clustering we will use a generated (synthetic) data set. The set is downloaded from
- http://cs.joensuu.fi/sipu/datasets/

K-means Clustering

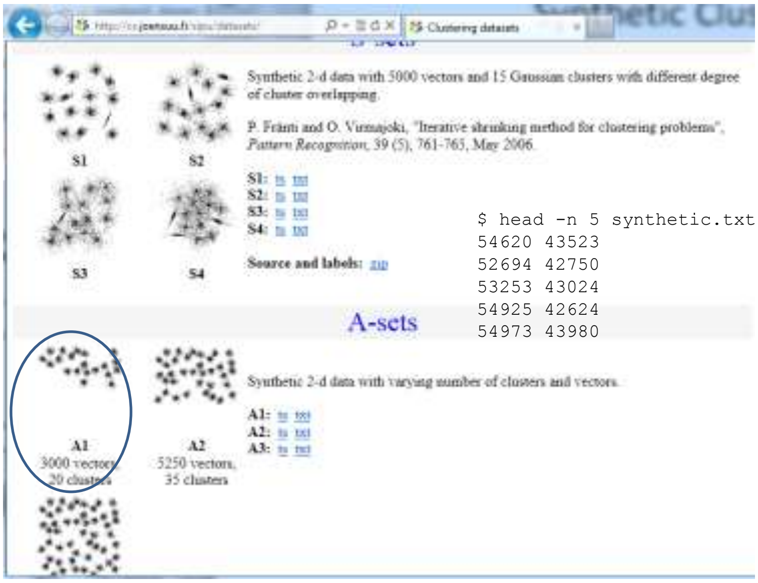
- K-means is the oldest and simplest clustering algorithm. With K-means you tell the K-means algorithm ahead of time how many clusters you're looking for (the *K* in K-means).
- The K-means process starts with the initial placement of the *K* cluster centroids.
- The initial K centroids can either be randomly located or specifically placed. Randomly located centroids will likely give different results, and therefore the recommendation is that centroids are located as far away from each other as possible.
- Once the initial centroids have been placed, K-means follows an iterative algorithm whereby each data point is associated to the nearest cluster centroid, and then the cluster centroids are repositioned relative to all the data points. This process repeats until such a time as the cluster centroids don't move, at which time the clusters are considered to have converged.
- To determine the distances between data points and the cluster centroids, clustering supports most of the similarity metrics that you saw in the recommenders section, such as Euclidean distance, Tanimoto, and Manhattan.
- A high-level algorithm for K-means is as follows:

1. Model the objects that you want to cluster into *N* dimensions.
2. Place the *K* centroids into the space represented by the objects.
3. Using a distance metric, assign each object to the centroid that is closest to it.
4. Recalculate the position of the *K* centroids.
5. Repeat steps 3 and 4 until either (a) the *K* centroids don't move/converge beyond a certain threshold, or (b) the maximum number of iterations has been reached

Sequence of Iterations in K-means


1 Model the data that you want to cluster
2 Place the K centroids
Initial centroids
3 Assign each object to the closest centroid
4 Recalculate the positions of the K centroids
Repeat steps 3 and 4 until the centroids don't move
5 Assign each object to the closest centroid
Object 4 is assigned to centroid A because it is closer
6 Recalculate the positions of the K centroids

Synthetic Clustering Dataset



Synthetic 2-d data with 5000 vectors and 15 Gaussian clusters with different degree of cluster overlapping.

P. Fränti and O. Virmajoki, "Iterative shrinking method for clustering problems", *Pattern Recognition*, 39 (5), 761-765, May 2006.

S1 S2 S3 S4

S1: ts txt
S2: ts txt
S3: ts txt
S4: ts txt

Source and labels: zip

A-sets

Synthetic 2-d data with varying number of clusters and vectors.

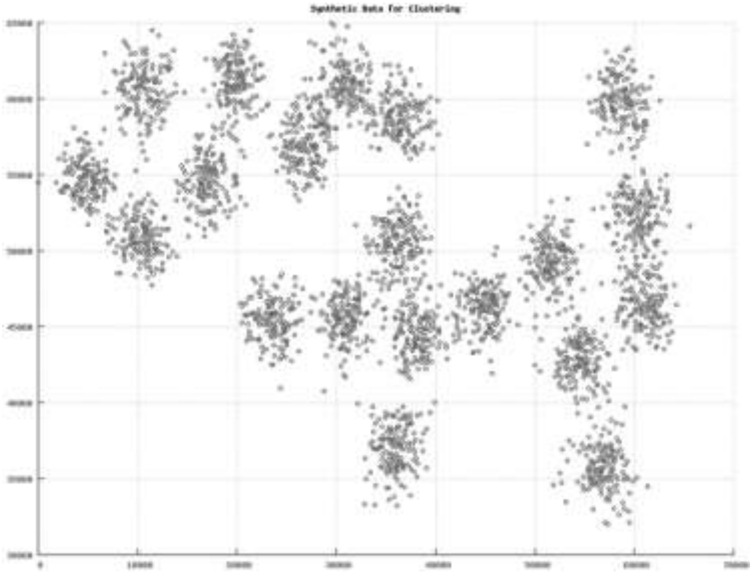
A1: ts txt
A2: ts txt
A3: ts txt

A1
3000 vectors,
20 clusters

A2
5250 vectors,
35 clusters

```
$ head -n 5 synthetic.txt
54620 43523
52694 42750
53253 43024
54925 42624
54973 43980
```

Scatter Plot 3,000 2D data points



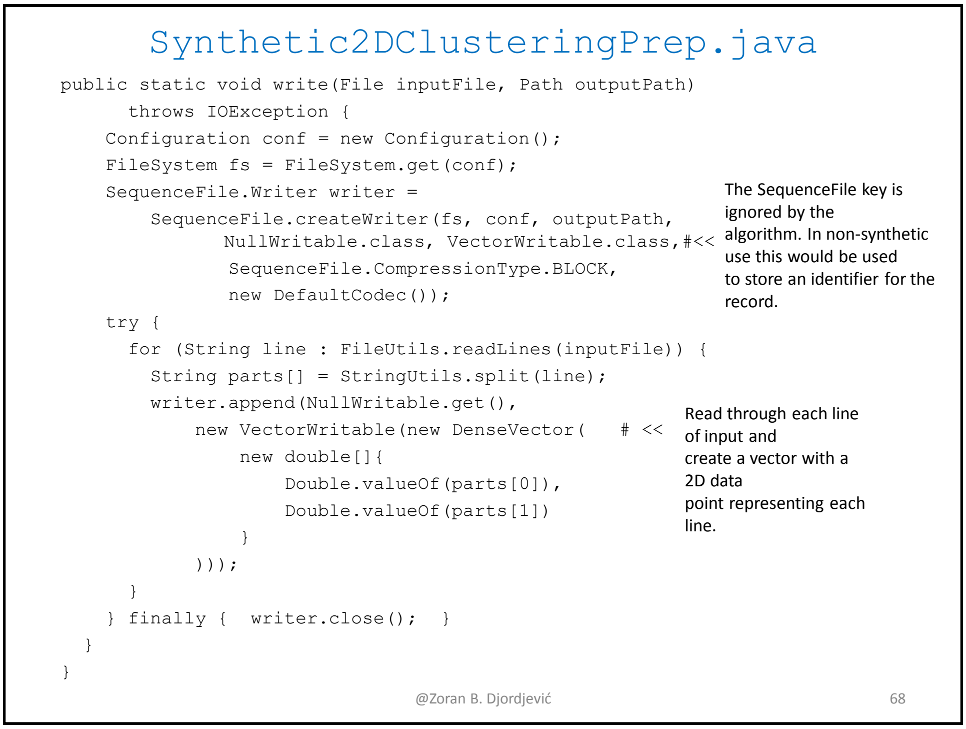
SequenceFile format

- Mahout for clustering requires as its input data in the SequenceFile format.
- We have little choice. We need to convert the synthetic 2D data into the required format.

Synthetic2DClusteringPrep.java

```
package edu.hu.mahout;
import org.apache.commons.io.FileUtils;
import org.apache.commons.lang.StringUtils;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.SequenceFile;
import org.apache.hadoop.io.compress.DefaultCodec;
import org.apache.mahout.math.DenseVector;
import org.apache.mahout.math.VectorWritable;
import java.io.File;
import java.io.IOException;

public class Synthetic2DClusteringPrep {
  public static void main(String... args) throws IOException {
    write(new File(args[0]), new Path(args[1]));
  }
```

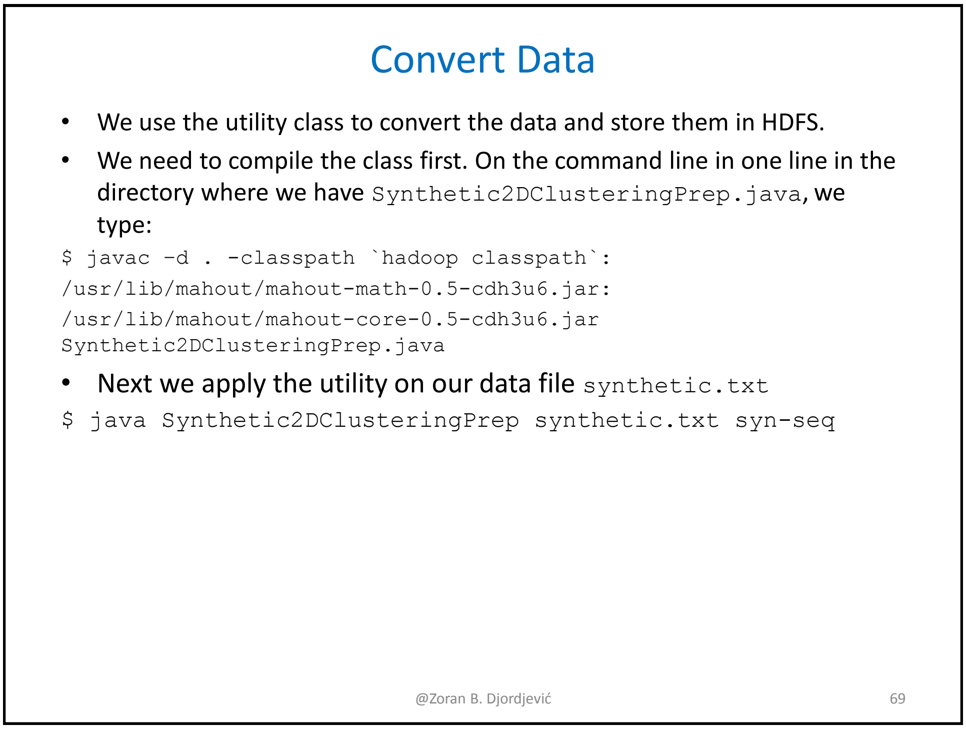


Synthetic2DClusteringPrep.java

```
public static void write(File inputFile, Path outputPath)
      throws IOException {
    Configuration conf = new Configuration();
    FileSystem fs = FileSystem.get(conf);
    SequenceFile.Writer writer =
        SequenceFile.createWriter(fs, conf, outputPath,
               NullWritable.class, VectorWritable.class,#<<
               SequenceFile.CompressionType.BLOCK,
               new DefaultCodec());
    try {
      for (String line : FileUtils.readLines(inputFile)) {
        String parts[] = StringUtils.split(line);
        writer.append(NullWritable.get(),
            new VectorWritable(new DenseVector(   # <<
                new double[]{
                    Double.valueOf(parts[0]),
                    Double.valueOf(parts[1])
                }
            )));
      }
    } finally {  writer.close();  }
  }
}
```

The SequenceFile key is ignored by the algorithm. In non-synthetic use this would be used to store an identifier for the record.

Read through each line of input and create a vector with a 2D data point representing each line.



Convert Data

- We use the utility class to convert the data and store them in HDFS.
- We need to compile the class first. On the command line in one line in the directory where we have `Synthetic2DClusteringPrep.java`, we type:

```
$ javac -d . -classpath `hadoop classpath`:
/usr/lib/mahout/mahout-math-0.5-cdh3u6.jar:
/usr/lib/mahout/mahout-core-0.5-cdh3u6.jar
Synthetic2DClusteringPrep.java
```

- Next we apply the utility on our data file `synthetic.txt`

```
$ java Synthetic2DClusteringPrep synthetic.txt syn-seq
```

Run the Clustering

- Create an empty HDFS directory

```
$ hadoop fs -mkdir syn-clusters

$ mahout kmeans \
-i syn-seq \         # input directory in HDFS with SequenceFile
-c syn-clusters \    # the path to initial centroids, empty now
-o syn-kmeans \      # the output working directory
-dm org.apache.mahout.common.distance.EuclideanDistanceMeasure \
-x 100 \              # number of iterations
-k 20 \               # number of clusters
-ow \                 # overwrite output directory if it exists
--clustering
```

- This will start an iterative sequence of MapReduce jobs until such a time as the clusters converge, or you hit the maximum number of iterations.
- When the clustering has completed there should be a number of directories in HDFS that contain output for each of the MapReduce iterations.

Summary

- When the clustering has completed, you can use the clusterdump Mahout utility to dump out the cluster details of the last job:

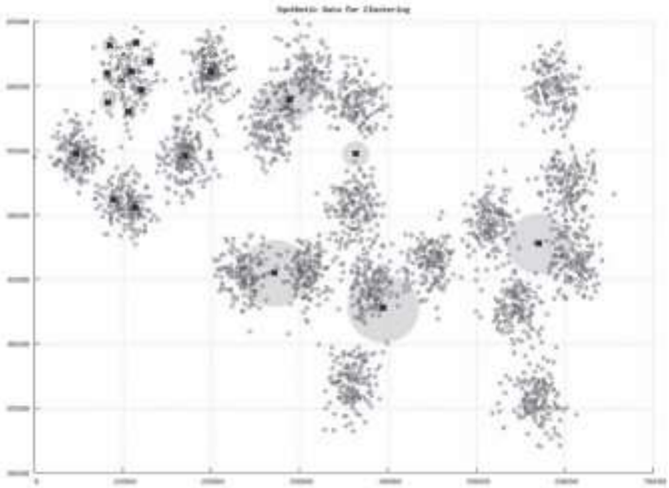
```
$ mahout clusterdump -s syn-kmeans/clusters-22-final
```

- `22` is the number of the last iteration
- `clusterdump` writes out a line for each cluster. `VL` indicates that the cluster has converged, and `CL` means that the cluster hasn't converged.

```
    VL-2976{
      n=65                    # number of points connected to the cluster
      c=[8906.923, 51193.292]  # cluster centroid
      r=[955.286, 1163.688]    # The radius of the cluster, expressed
}                              # as a standard deviation of the
                               # distance from the centroid to all 65
                                 data points in the cluster
VL-2997{n=88 c=[11394.705, 50557.114] r=[920.032, 1179.291]}
VL-2950{n=464 c=[39502.394, 42808.983] r=[4022.406, 4273.647]}
VL-2956{n=900 c=[57117.122, 47795.646] r=[3623.267, 7669.076]}
VL-2963{n=307 c=[28842.176, 58910.573] r=[2532.197, 2463.770]}
VL-2968{n=24 c=[12087.458, 59659.125] r=[610.980, 587.461]}
VL-2973{n=21 c=[9767.762, 60524.619] r=[334.271, 680.851]}
VL-2974{n=149 c=[17056.611, 54574.094] r=[1424.306, 1499.089]}
```

How did we do

- The black crosses are the cluster centroids after the clustering has completed.



- The solid gray circle represents the cluster radius standard deviation from the centroid to all points in the cluster.

Other Mahout Clustering Algorithms

Algorithm	Description
Hierarchical clustering/ Top Down clustering	Hierarchical clustering is the process or finding bigger clusters, and also the smaller clusters inside the bigger clusters. Top Down clustering is a type of hierarchical clustering. It tries to find bigger clusters first and then does finegrained clustering on these clusters—hence the name *Top Down*.
Canopy clustering	Canopy clustering is a simple, fast, and surprisingly accurate method for grouping objects into clusters. Canopy clustering is often used as an initial step in more rigorous clustering techniques, such as K-means clustering. By starting with an initial clustering, the number of more expensive distance measurements can be significantly reduced by ignoring points outside of the initial canopies.
Fuzzy K-means	Fuzzy K-means (also called Fuzzy C-means) is an extension of K-means, the popular simple clustering technique. While K-means discovers hard clusters (a point belong to only one cluster), Fuzzy K-means is a more statistically formalized method and discovers soft clusters where a particular point can belong to more than one cluster with certain probability.
Latent Dirichlet Allocation (LDA)	Latent Dirichlet Allocation (Blei et al., http://www.cs.princeton.edu/~blei/papers/BleiNgJordan2003.pdf, 2003) is a powerful learning algorithm for automatically and jointly clustering words into *topics* and documents into mixtures of topics.