

Recommender Systems

Daniel Rodriguez

University of Alcala



Some slides and examples based on Chapter 9,
Mining of Massive Datasets, Rajaraman et al., 2011
(DRAFT)

Outline

- 1 Introduction
- 2 Recommender Systems
- 3 Algorithms
- 4 Evaluation of Recommender Systems
- 5 Dimensionality Reduction
- 6 Tools
- 7 Problems and Challenges
- 8 Conclusions
- 9 References

- 1 Introduction
- 2 Recommender Systems
- 3 Algorithms
- 4 Evaluation of Recommender Systems
- 5 Dimensionality Reduction
- 6 Tools
- 7 Problems and Challenges
- 8 Conclusions
- 9 References

Recommender systems

Definition (by Ricci et al. [10])

Recommender Systems (RSs) are software tools and techniques providing suggestions for items to be of use to a user.

The suggestions relate to various decision-making processes, such as what items to buy, what music to listen to, or what online news to read.

Item is the general term used to denote what the system recommends to users.

A RS normally focuses on a specific type of item (e.g., CDs, or news) and accordingly its design, its graphical user interface, and the core recommendation technique used to generate the recommendations are all customized to provide useful and effective suggestions for that specific type of item.

Recommender systems

http://en.wikipedia.org/wiki/Recommender_system

Recommender systems (RS) are a subclass of information filtering system that seek to predict the 'rating' or 'preference' that user would give to an item (such as music, books, or movies) or social element (e.g. people or groups) they had not yet considered, using a model built from the characteristics of an item (content-based approaches) or the user's social environment (collaborative filtering approaches).

- The goal of these systems is to serve the right items to a user in a given context or create bundle packs of similar products to optimize long term business objectives
- Foundations based on data mining, information retrieval, statistics, etc.
- Established area of research with an ACM Conference:
 - <http://recsys.acm.org/>

Recommender systems: Applications

- Product Recommendations: Perhaps the most important use of recommendation systems is at on-line retailers, etc., e.g, Amazon or similar on-line vendors strive to present users suggestions of products that they might like to buy. These suggestions are not random, but based on decisions made by similar customers.
- News Articles: News services usually classify interesting news for some people to offer them to similar users. The similarity might be based on the similarity of important words in the documents, or on the articles that are read by people with similar reading tastes. (the same principles apply to recommending blogs, videos, etc.)
- Movie/Music Recommendations, e.g. Netflix offers customers recommendations of movies they might be interested in. These recommendations are based on ratings provided by users.
- Application stores
- Social Networks
- etc.

Recommender systems: Examples

Recommendations for You in PC & Video Games



Battlefield 3 (PS3)
Electronic Arts
PlayStation 3
★★★★☆ (181)
£17.95
[Fix this recommendation](#)



Need for Speed Most Wanted (PS3)
Electronic Arts
PlayStation 3
★★★★☆ (139)
£12.99
[Fix this recommendation](#)



Need for Speed: The Run (PS3)
Electronic Arts
PlayStation 3
★★★★☆ (76)
£11.98
[Fix this recommendation](#)



Sony PlayStation 3 12GB Super Slim...
PlayStation 3
★★★★☆ (124)
£155.79
[Fix this recommendation](#)



Need For Speed: Hot Pursuit (PS3)
Electronic Arts
PlayStation 3
★★★★☆ (207)
£11.40
[Fix this recommendation](#)



WRC 3 - World Rally Championship (PS3)
pauze
PlayStation 3
★★★★☆ (41)
£18.68
[Fix this recommendation](#)

[See more recommendations](#)

Figure: Amazon Recommendations

Recommender systems: Examples

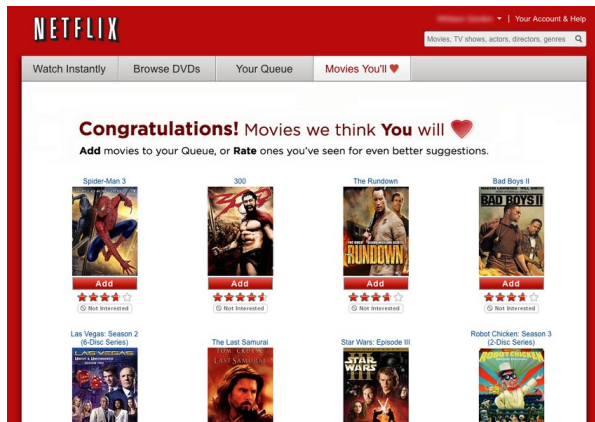



Figure: Netflix recommendations

Recommender systems: Examples

Review by Pitchfork ★★★★★

Since you follow **Kylie Minogue**, we recommend **X** for your collection.




X
Kylie Minogue

Save as a Playlist

The best summary of Kylie Minogue comes from Grant Morrison's 1992 Zenith comic: "Kylie is Vera Lynn for Third World War,"

Review by Pitchfork ★★★★★

You listened to **Toni Braxton**. Here's an album you might like.



BLACKsummers'night
Maxwell

Save as a Playlist

Heartbreak is a constant in popular music, and with good reason; Maxwell is among the billions across the globe who have had the

Figure: Spotify recommendation

Recommender systems: Examples

The screenshot shows a web browser window displaying a ScienceDirect article page. The article title is "An empirical study of maintenance and development estimation accuracy". A modal window titled "ScienceDirect article suggestions" is overlaid on the page. This window contains a section for "Related articles" which lists three articles with their titles, journal information, and authors. Each article entry includes a "Show abstract" link and a PDF icon with the file size. At the bottom of the modal, there is a link to "View more related articles" and a "Do not show again" button.

ScienceDirect article suggestions

Related articles

These articles have key terms similar to those in the article you downloaded. [Learn more](#)

- On the problem of the software cost function**
Information and Software Technology, Volume 43, Issue 1, 1 January 2001, Pages 61-72
J.J. Dolado
[Show abstract](#) | [PDF \(199 K\)](#)
- A new calibration for Function Point complexity weights**
Information and Software Technology, Volume 50, Issues 7-8, June 2008, Pages 670-683
Wei Xia, Luiz Fernando Capretz, Danny Ho, Fahiem Ahmad
[Show abstract](#) | [PDF \(777 K\)](#)
- Ensembles and locality: Insight on improving software effort estimation**
Information and Software Technology, Volume 55, Issue 8, August 2013, Pages 1512-1528
Leandro L. Minku, Xin Yao
[Show abstract](#) | [PDF \(391 K\)](#)

[View more related articles](#)

[Do not show again](#)

Long Tail Phenomenon

Physical institutions can only provide what is popular (shelf space is scarce), while on-line institutions can make everything available [2]. The greater the choice, the greater need for better filters.

Long Tail Phenomenon

Recommender systems take advantage of large amounts of data available on the Internet to create recommendations that in “bricks and mortar” stores would be impossible to do.

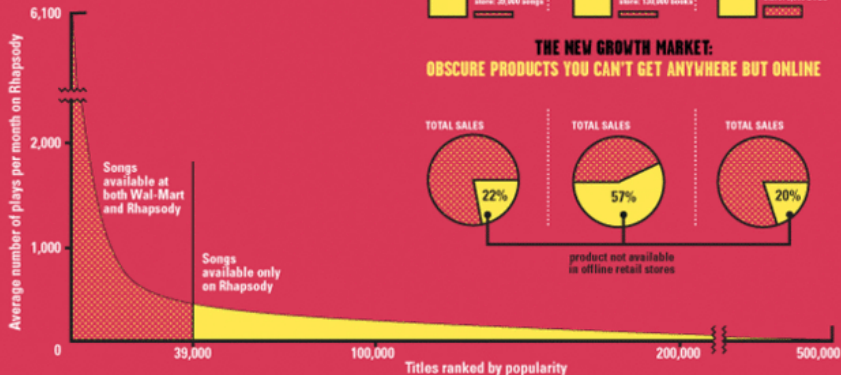
This phenomenon is called “long tail phenomenon” and explains the impossibility of the physical stores to create personalized recommendations.



Long Tail Phenomenon

ANATOMY OF THE LONG TAIL

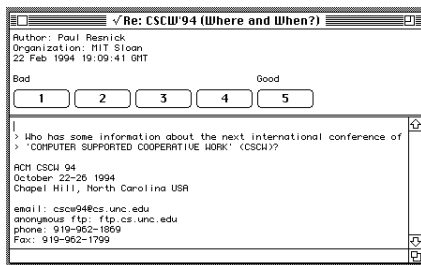
Online services carry far more inventory than traditional retailers. Rhapsody, for example, offers 19 times as many songs as Wal-Mart's stock of 39,000 tunes. The appetite for Rhapsody's more obscure tunes (charted below in yellow) makes up the so-called Long Tail. Meanwhile, even as consumers flock to mainstream books, music, and films (right), there is real demand for niche fare found only online.



Sources: Erik Brynjolfsson and Jeffrey Hu, MIT, and Michael Smith, Carnegie Mellon; Barnes & Noble; Netflix; RealNetworks

Recommender systems: History

The first recommender system, Tapestry, was designed to recommend documents from newsgroups. The authors also introduced the term *collaborative filtering* as they used social collaboration to help users with large volume of documents.



P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, J. Riedl, GroupLens: An Open Architecture for Collaborative Filtering of Netnews, Proc. of Computer Supported Cooperative Work (CSCW), pp. 175-186, 1994¹

¹<http://ccs.mit.edu/papers/CCSWP165.html>

The Netflix Prize

The importance of predicting ratings accurately is so high that on Oct 2006, Netflix, decided to create a contest to try to improve its recommendation algorithm, it was the Netflix Prize:

<http://www.netflixprize.com/>



Netflix offered **\$1m** to the team capable of proposing an algorithm 10% better than their current algorithm (CineMatch). To so so, Netflix published a dataset with:

- approximately 17,000 movies
- 500,000 users
- 100 million ratings (training set of 99 million ratings)

The Root-mean-square error (RMSE) was used to measure the performance of algorithms. CineMatch had an RMSE of 0.9525.

The Netflix Prize: Results

The contest finished almost 3 years later and the winner improved the algorithm in 10.6%. The winner, a team of researchers called “Bellkor’s Pragmatic Chaos.

They must also share the algorithm with Netflix and the source code. In this contest participated more than 40,000 teams from 186 different countries.

The winning algorithm was a composition of different algorithms that had been developed independently.

Also Moshfeghi [7] made another approach adding semantics to the algorithms to improve the accuracy.



Netflix Prize

COMPLETED

Characteristics

A recommender system must be reliable providing good recommendations and showing information about the recommendations (explanations, details, etc.). Another important point of these systems is how they should display the information about the recommended products:

- The item recommended must be easy to identify by the user.
- Also the item must be easy to evaluate/correct (“I don’t like it”, “I already have it”, etc.).
- The ratings must be easy to understand and meaningful.
- Explanations must provide a quick way for the user to evaluate the recommendation.

Recommendations for You in PC & Video Games



Battlefield 3 (PS3)
Electronic Arts
PlayStation 3
★★★★☆ (281)
£17.95

[Fix this recommendation](#)

[See more recommendations](#)



Need for Speed Most Wanted (PS3)
Electronic Arts
PlayStation 3
★★★★☆ (119)
£12.99

[Fix this recommendation](#)



Need for Speed: The Run (PS3)
Electronic Arts
PlayStation 3
★★★★☆ (76)
£11.98

[Fix this recommendation](#)



Sony PlayStation 3 12GB Super Slim...
PlayStation 3
★★★★☆ (124)
£155.79

[Fix this recommendation](#)



Need For Speed: Hot Pursuit (PS3)
Electronic Arts
PlayStation 3
★★★★☆ (107)
£11.40

[Fix this recommendation](#)



WRC 3 - World Rally Championship (PS3)
pslogic
PlayStation 3
★★★★☆ (41)
£18.68

[Fix this recommendation](#)

Characteristics: Degree of Personalisation

The degree of personalisation of this recommendations can be different for each site. Galland [4] classified the recommendations into 4 groups:

- Generic: everyone receives same recommendations.
- Demographic: everyone in the same category receives same recommendations.
- Contextual: recommendation depends only on current activity.
- Persistent: recommendation depends on long-term interests.

Other Characteristics

A recommender system usually involves [1]:

- Large scale Machine Learning and Statistics
- Off-line Models (capture global and stable characteristics)
- On-line Models (incorporates dynamic components)
- Explore/Exploit (active and adaptive experimentation) – Multi-Objective Optimization
- Click-rates (CTR), Engagement, advertising revenue, diversity, etc. – Inferring user interest
- Constructing User Profiles - Natural Language Processing to understand content.
- Topics, “aboutness”, entities, follow-up of something, breaking news, etc.

- 1 Introduction
- 2 Recommender Systems**
- 3 Algorithms
- 4 Evaluation of Recommender Systems
- 5 Dimensionality Reduction
- 6 Tools
- 7 Problems and Challenges
- 8 Conclusions
- 9 References

Utility Matrix

Recommender Systems are often seen as a function:

- $C \in \text{Customers}$
- $I \in \text{Items}$
- $R \in \text{Ratings}$, e.g., [1-5], [+,-], [a+,E-], etc.
- Utility function: $u : C \times I \rightarrow R$

	I_1	I_2	\dots	I_n	
U_1	4	?	1	5	6
U_2		5	7	6	
\dots					
U_n		3			3

Blanks correspond to not rated items.

Objective: would U_1 like I_2 ?

Sparse Matrices

Typically most of the values are empty, these matrices are called **sparse matrices**.

- Users rate just a few percentage of all available items
- An unknown rating implies that we have no explicit information about the user's preference for that particular item.

The **goal of a RS is to predict the *blanks*** (fill the gaps) in the matrix.

- Generally, it is not necessary to fill up all of the blanks because we usually are not interested in low rated items
- The majority of RS recommend a few of the highest rated items

Sparse Matrices - Representation

In order to fit in memory, zero values are not explicitly represented. For instance, in ARFF format (Weka)

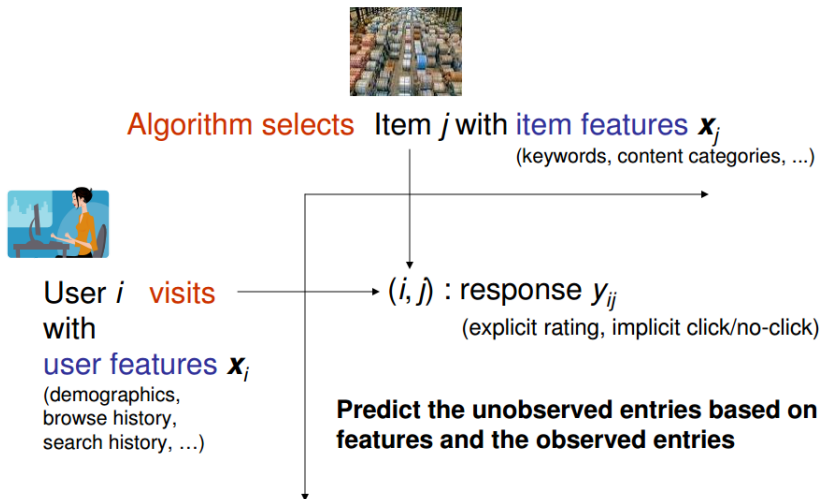
```
0, X, 0, Y, "class A"  
0, 0, W, 0, "class B"
```

is represented by their attribute number and value as:

```
{1 X, 3 Y, 4 "class A"}  
{2 W, 4 "class B"}
```

Work-flow in recommendations

There are a lot of different ways to get recommendations but the most used are based on the previous knowledge of similar users or contents [5].



Naïve Recommender Systems

- Editorial recommendations
- Simple aggregates
 - Top 10, Most Popular, etc.

From now on, we will refer to systems tailored to individual users (Amazon, Netflix, etc.)

Recommender Systems: Classification

Typically, also in Rajamaran et al. [8], recommender systems are classified according to the technique used to create the recommendation (fill the blanks in the utility matrix):

- **Content-based** systems examine properties of the items recommended and offer similar items
- **Collaborative filtering (CF)** systems recommend items based on similarity measures between users and/or items. The items recommended to a user are those preferred by similar users
- **Hybrid** mixing both previous approaches

Recommender Systems: Classification

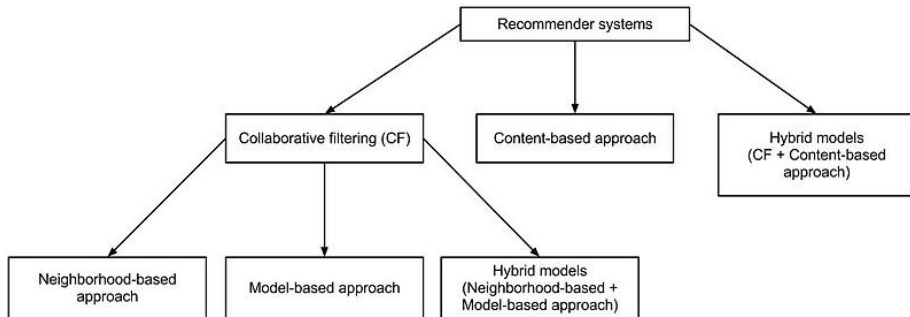


Figure: Classification Recommender Systems [8]

Recommender Systems: Content-based

- **Content-based** systems examine properties of the items to recommend items that are similar in content to items the user has already liked in the past, or matched to attributes of the user.
- For instance, movie recommendations with the same actors, director, genres, etc., if a Netflix user has watched many action movies, then recommend movies classified in the database as having the “action” genre.
- Textual content (news, blogs, etc) recommend other sites, blogs, news with similar content (we will cover how to measure similar content)

Item Profiles

- For each item, we need to create an item profile
- A profile is a set of features
 - Context specific (e.g. with films: actors, director, genre, title, etc.)
 - Documents: sets of important words. Important words are usually selected using the is $TF.IDF$ (Term Frequency times Inverse Doc Frequency) metric

Term Frequency-Inverse Document Frequency

The Term Frequency-Inverse Document Frequency (**TF-IDF**) is a statistic which reflects how important a word is to a document². The profile of a document is the set of words with highest $tf - idf$, which assigns a weight to the term t in a document d .

$$tf.idf_{d,j} = tf_{t,d} \cdot idf_t \quad (1)$$

where

- $tf_{t,d}$ no. of times term t occurs in document d (there are better approaches)
- $idf_t = \log \frac{N}{df_t}$, df_t is no. of documents that contain the term t ; and N is total no. of documents.

²<http://en.wikipedia.org/wiki/Tf-idf>

Tags

- Tags are also used to create item profiles. Then, tags are used to provide similar content.
For example, it is difficult to automatically extract features from images. In this case, users are asked to tag them.
- A real example is `del.icio.us` in which users tag Web pages:
 - `http://delicious.com/`
- The drawback of this approach is that it is difficult to get users to tag items.

User Profiles

We also need to create vectors with the same components that describe the user's preferences.

It is classified as implicit or explicit.

- Implicit refers to observe user's behaviour with the system, for example by watching certain films, listening to music, reading a kind of news or downloading applications/documents
- Explicit refers when the user provides information to the system

Collaborative Filtering

In Collaborative Filtering (CF), a user is recommended items based on the past ratings of all users collectively. CF can be of two types:

- User-based collaborative filtering
 - ① Given a user U , find the set of other users D whose ratings are similar to the ratings of the selected user U .
 - ② Use the ratings from those like-minded users found in Step 1 to calculate a prediction for the selected user U .
- Item-based collaborative filtering
 - ① Build an item-item matrix determining relationships between pairs of items
 - ② Using this matrix and data on the current user, infer the user's taste

Hybrid Methods

- Implement two separate recommenders and combine predictions
- Add content-based methods to collaborative filtering
 - Item profiles for new item problem
 - Demographics to deal with new user problem

Ratings

Rating

A numeric (usually) value given by a user to specific item/user [10].

The way we populate the ratings matrix is also very important. However acquiring data from which to build a ratings matrix is often a very difficult step. There are two general approaches to discover the rating value of the users:

- Ask directly to the user to insert ratings. This approach is typically used by the movie sites (Netflix) but is limited by the willing of the users to rate items.
- “Spy” the behaviour of the user. If a user watches a movie, reads an article or buys an item we can say that the user likes such particular item. In this case, the ratings matrix will be filled with the value 0 if the user does not buy/watch/read the item and 1 if the user buy/watch/read it.

Ratings — Classification

Implicit ratings

- Based on interaction & time: Purchases, clicks, browsing (page view time), etc.
- Used to generate an implicit numeric rating.

Explicit ratings

- Numeric ratings: numeric scale between 2 (+/-) and 15 values (the more levels, the more variance; should be normalized)
- Partial order: comparison between two items.
- Semantic information: tags, labels.

Hybrid

Mixing both previous approaches.

Measuring Similarity

One of the more complicated task is to determine is the similarity between the users/items (the duality of similarity [8]).

Following the example by Rajaraman and Ullman [8], we can observe that users *A* and *C* both rated items *TW1* and *SW1* totally different.

Intuitively we could say that there is a large distance regarding their similarity. We next explain different methods that can be used to check the similarity between users.

	<i>HP</i> ₁	<i>HP</i> ₂	<i>HP</i> ₃	<i>TW1</i>	<i>SW1</i>	<i>SW2</i>	<i>SW3</i>
<i>A</i>	4			5	1		
<i>B</i>	5	5	4				
<i>C</i>				2	4	5	
<i>D</i>		3					3

Table: Utility Matrix [8]

Jaccard Distance

The **Jaccard index** (aka Jaccard similarity coefficient) measures the similarity of two sets³:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \quad (2)$$

The Jaccard distance measures the dissimilarity and is defined as the complement of Eq(2):

$$J_d(A, B) = 1 - J(A, B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|} \quad (3)$$

The Jaccard distance only takes into account the number of rated items but not the actual rating which is discarded, therefore, it loses accuracy in case that detailed recommendations were needed.

³http://en.wikipedia.org/wiki/Jaccard_index

Jaccard Distance

Following with the previous example [8]:

	HP_1	HP_2	HP_3	$TW1$	$SW1$	$SW2$	$SW3$
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

A and B have an intersection of size 1, i.e., $|A \cap B| = 1$ and a union of size 5, $|A \cup B| = 5$ (items rated). Thus, their Jaccard similarity is $1/5$, and their Jaccard distance is $4/5$; i.e., they are very far apart.

A and C have a Jaccard similarity of $2/4$, so their Jaccard distance is the same, $1/2$.

However, although A appears to be closer to C than to B , that conclusion seems intuitively wrong. A and C disagree on the two movies they both watched, while A and B seem both to have liked the one movie they watched in common [8].

Cosine Distance

Cosine Distance:

- Items are represented as vectors over the user space
- Similarity is the cosine of the angle between two vectors
- Range is between 1 (perfect) and -1 (opposite)

Given two vectors of attributes, A and B , the cosine similarity, $\cos(\theta)$, is represented using a dot product and magnitude as⁴:

$$\cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^n A_i \times B_i}{\sqrt{\sum_{i=1}^n (A_i)^2} \times \sqrt{\sum_{i=1}^n (B_i)^2}} \quad (4)$$

In information retrieval, the cosine similarity of two documents will range from 0 to 1, since the term frequencies (*tf* – *idf* weights) cannot be negative. The angle between two term frequency vectors cannot be greater than 90 degrees.

⁴http://en.wikipedia.org/wiki/Cosine_similarity

Cosine Distance: Example

Following with the previous example [8], the Cosine distance between users A and B is:

$$\frac{4 \cdot 5}{\sqrt{4^2 + 5^2 + 1^2} \cdot \sqrt{4^2 + 5^2 + 5^2}} = 0.380 \quad (5)$$

The Cosine distance between A and C is:

$$\frac{5 \cdot 2 + 1 \cdot 4}{\sqrt{4^2 + 5^2 + 1^2} \cdot \sqrt{2^2 + 4^2 + 5^2}} = 0.322 \quad (6)$$

A larger (positive) cosine implies a smaller angle and therefore a smaller distance, this measure tells us that A is slightly closer to B than to C [8]. Empty values as set to 0 (questionable election as it could seem that users do not like the item instead of no rating form the user)

Pearson correlation

Another common measure of similarity is the Pearson correlation coef [U+FB01]cient between the ratings of the two users, a and u :

$$Pearson_{a,u} = \frac{\sum_{i \in I} (r_{a,i} - \bar{r}_a)(r_{u,i} - \bar{r}_u)}{\sqrt{\sum_{i \in I} ((r_{a,i} - \bar{r}_a)^2) \sum_{i \in I} ((r_{u,i} - \bar{r}_u)^2)}} \quad (7)$$

where:

- I is the set of items rated by both users
- $r_{u,i}$ is the rating of given to item i by user u
- \bar{r}_a, \bar{r}_u are the mean ratings given by users a and u respectively

Rounding Data

Rounding data is more like a pre-processing step of the information before applying any distance measure.

Following the example, we could consider ratings of 3, 4, and 5 as a “1” and consider ratings 1 and 2 as unrated (blanks).

	HP_1	HP_2	HP_3	$TW1$	$SW1$	$SW2$	$SW3$
A	1			1	1		
B	1	1	1				
C					1	1	
D		1					1

The Jaccard distance between A and B is $3/4$, while between A and C is 1. Now, C appears further away from A than B does, which is intuitively correct. Applying cosine distance allows us to reach the same conclusion.

Normalizing Ratings

As with the rounding data method, this is also a preprocessing step applied before any other measure distance.

Normalised ratings are calculated subtracting the average value of the ratings of the user to each single rating. Then low ratings will be negative and high ratings positive.

Following the example, the normalised rating matrix will be as follows:

	HP_1	HP_2	HP_3	$TW1$	$SW1$	$SW2$	$SW3$
A	$2/3$			$5/3$	$-7/3$		
B	$1/3$	$1/3$	$-2/3$				
C				$-5/3$	$1/3$	$4/3$	
D		0					0

Normalizing Ratings: Example

Applying the Jaccard distance to this normalised matrix, it is noted that ratings from the user D are “0”. The ratings given by the user D are (probably) not interesting (as this user has always rated items with the same value).

The Cosine distance in this new case for users A and B :

$$\frac{(2/3) \cdot (1/3)}{\sqrt{(\frac{2}{3})^2 + (\frac{5}{3})^2 + (-\frac{7}{3})^2} \sqrt{(\frac{1}{3})^2 + (\frac{1}{3})^2 + (-\frac{2}{3})^2}} = 0.092 \quad (8)$$

Now, the Cosine distance for the users A and C with normalized values:

$$\frac{(5/3) \cdot (-5/3) + (-7/3) \cdot (1/3)}{\sqrt{(\frac{5}{3})^2 + (\frac{5}{3})^2 + (-\frac{7}{3})^2} \sqrt{(-\frac{5}{3})^2 + (\frac{1}{3})^2 + (\frac{4}{3})^2}} = -0.559 \quad (9)$$

A and C are much further apart than A and B . This result makes sense because A and C rated 2 films with very different values and A and B rated only one film with similar values.

- 1 Introduction
- 2 Recommender Systems
- 3 Algorithms**
- 4 Evaluation of Recommender Systems
- 5 Dimensionality Reduction
- 6 Tools
- 7 Problems and Challenges
- 8 Conclusions
- 9 References

Knowledge Discovery in Databases (KDD)

RSs, natural language processing (NLP) are based on a subfield of computer science that tries to discover patterns in large data sets, called Data mining or Knowledge Discovery in Databases (KDD).

In fact, *data mining* is typically one step within the process of what is known as Knowledge Discovery in Databases (KDD).

The main goal of the data mining (the analysis step of the Knowledge Discovery in Databases (KDD) process [3]) is to extract information from a data set and transform it into an understandable structure for further use. This transformation involves several steps that includes: database and data management aspects, data pre-processing, model and inference considerations, interestingness metrics, complexity considerations, post-processing of discovered structures, visualization, and on-line updating. The step which requires most of the time is the preparation of data.

KDD Process

KDD is composed of the following steps [3]:

- 1 Selection
- 2 Pre-processing
- 3 Transformation
- 4 Data Mining
- 5 Interpretation/Evaluation

KDD Process

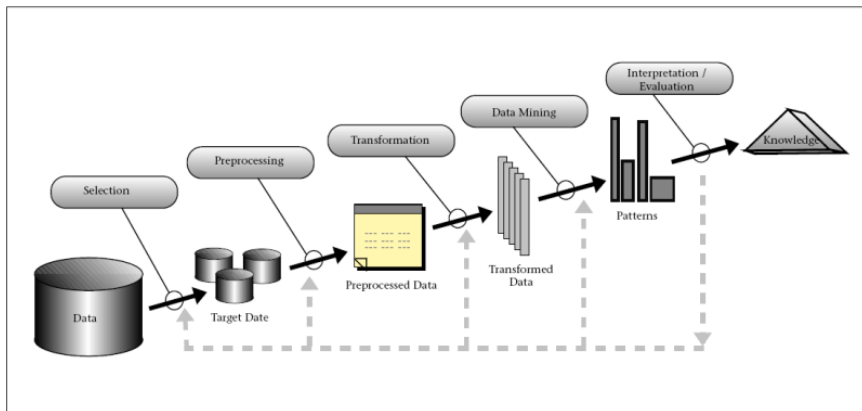


Figure: Data mining steps (Fayyad et al, 96)

Text Mining

Text mining, aka “text analytics” or Information Retrieval tries to obtain usable information for a computer using no structured (no measurable) data.

The Oxford Dictionary defines text mining as “process or practice of examining large collections of written resources in order to generate new information, typically using specialized computer software”.

Typically no measurable data refers to e-mails, newspapers, research papers, etc.

The process of text mining usually is divided in 4 different steps

- 1 Information retrieval (IR) systems.
- 2 Natural language processing (NLP).
- 3 Information extraction (IE).
- 4 Data mining (DM).

Web Mining

Web mining uses data mining tools to extract information from:

- Web pages (Web content mining)
- Links (Web structure mining)
- User's navigation data (Web usage mining) based on Web server logs

Nearest Neighbour Algorithm

The Nearest Neighbour (k -NN) algorithm is one of the simplest machine learning algorithms but works very well in practice⁵.

The idea is to predict a classification based on the k -nearest neighbours of the item we want to classify.

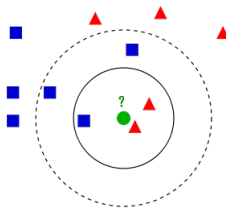
No model is in fact learned, but the active user is used to search for the k most similar cases

⁵http://en.wikipedia.org/wiki/K-nearest_neighbor_algorithm 🔍

Nearest Neighbour Algorithm

A simple approximation of k NN is shown in Figure where we need to classify the green circle:

- 1 If we select $k = 3$, the nearest neighbours are 2 red triangles and a blue square, represented inside the solid black circle, the class of the circle will be red triangle.
- 2 In the case of $k = 5$ (discontinuous black circle), the nearest neighbours are 2 red triangles and 3 blue squares, then the class of the circle will be blue square.



The selection of k is very important (it can change the class)

kNN: Complexity

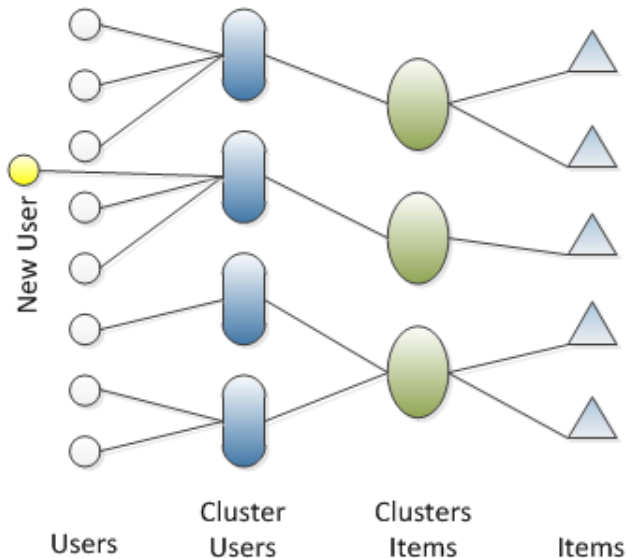
- 1 Expensive step is finding k most similar customers Too expensive to do at runtime, need to pre-compute
- 2 Can use clustering, partitioning as alternatives, but quality degrades

Clustering

With little data (ratings) or to reduce the matrix, clustering can be used.
A hierarchical clustering (until a desired level can be used).

	<i>HP</i>	<i>TW</i>	<i>SW</i>
<i>A</i>	4	5	1
<i>B</i>	4.67		
<i>C</i>		2	4.5
<i>D</i>	3		3

Clustering



Association Rules

Association Rules are a data mining technique to find relations between variables, initially used for shopping behaviour analysis.

Associations are represented as rules, *Antecedent* \rightarrow *Consequent*

- If a customer buys A, that customer also buys B
Association among A and B means that the presence of A in a record implies the presence of B in the same record
 $\{Nappies, Chips\} \rightarrow Beer$

Quality of the rules need a minimum support and confidence

- Support: the proportion of times that the rule applies
- Confidence: the proportion of times that the rule is correct

Slope One

This algorithm was introduced in 2005 and is the simplest form of item-based algorithm based on ratings. For a better understanding we are going to explain this algorithm with a little example.

First we need a dataset, for example, the next table shows the ratings from 3 people to 3 different items.

Customer	Item A	Item B	Item C
John	5	3	2
Mark	3	4	-
Lucy	-	2	5

Table: Example Slope One algorithm

Slope One

This algorithm is based on the average differences between users who rated the same items that we want to predict.

We want to know the rating that Lucy will give to the item A. Then we need to calculate the average differences of the other users with the item A and other item as reference (item B):

- John, item A rating 5, item B rating 3, difference 2
- Mark, item A rating 3, item B rating 4, difference -1
- Average difference between item A and item B, $(2+(-1))/2=0.5$
- Rating from Lucy to item B, 2
- Estimated rating from Lucy to item A, $2+0.5=2.5$

Slope One

Also we want to know the rating of Mark for the item *C*. We need to calculate the average differences of the other users with the item *C* and other item as reference (item *B*):

- John, item *C* rating 2, item *B* rating 3, difference -1
- Lucy, item *C* rating 5, item *B* rating 2, difference 3
- Average difference between item *C* and item *B*, $((-1)+3)/2=1$
- Rating from Mark to item *B*, 4
- Estimated rating from Mark to item *C*, $4+1=5$

Therefore, the final table of ratings is completed as follows:

Customer	Item A	Item B	Item C
John	5	3	2
Mark	3	4	5
Lucy	2.5	2	5

Decision Trees

A decision tree is a collection of nodes, arranged as a binary tree.

A decision tree is constructed in a top-down approach. The leaves of the tree correspond to classes (decisions such as “likes” or “dislikes”), Each interior node is a condition that correspond to features, and branches to their associated values.

To classify a new instance, one simply examines the features tested at the nodes of the tree and follows the branches corresponding to their observed values in the instance.

Upon reaching a leaf, the process terminates, and the class at the leaf is assigned to the instance.

The most popular decision tree algorithm is C4.5 which uses the gain ratio criterion to select the attribute to be at every node of the tree.

Naïve Bayes

The naïve Bayes algorithm uses the Bayes theorem to predict the class for each case, assuming that the predictive attributes are independent given a category.

A Bayesian classifier assigns a set of attributes A_1, A_2, \dots, A_n to a class C such that $P(C|A_1, A_2, \dots, A_n)$ is maximum, that is the probability of the class description value given the attribute instances, is maximal.

- 1 Introduction
- 2 Recommender Systems
- 3 Algorithms
- 4 Evaluation of Recommender Systems**
- 5 Dimensionality Reduction
- 6 Tools
- 7 Problems and Challenges
- 8 Conclusions
- 9 References

Evaluation Measures: Confusion Matrix

Table: Confusion Matrix for Two Classes

		Actual		
		Positive	Negative	
Pred	Positive	True Positive (TP)	False Positive (FP) Type I error (False alarm)	Positive Predictive Value (PPV)= Confidence = Precision = $= \frac{TP}{TP+FP}$
	Negative	False Negative (FN) Type II error	True Negative (TN)	Negative Predictive Value (NPV)= $\frac{TN}{FN+TN}$
		Recall = Sensitivity = $TP_r = \frac{TP}{TP+FN}$	Specificity = $TN_r = \frac{TN}{FP+TN}$	

Evaluation Measures using the confusion Matrix

- *True positive rate* ($TP/TP + FN$) is the proportion of positive cases correctly classified as belonging to the positive class.
- *False negative rate* ($FN/TP + FN$) is the proportion of positive cases misclassified as belonging to the negative class.
- *False positive rate* ($FP/FP + TN$) is the proportion of negative cases misclassified as belonging to the positive class.
- *True negative rate* ($TN/FP + TN$) is the proportion of negative cases correctly classified as belonging to the negative class.

There is a trade-off between *false positive rates* and *false negative rates* as the objective is to minimize both metrics (or conversely, maximize the true negative and positive rates). Both metrics can be combined to form single metrics. For example, the predictive *accuracy* is defined as:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (10)$$

Evaluation Measures using the confusion Matrix

Another metrics from the Information Retrieval field that is widely used when measuring the performance of classifiers is the *f-measure*, which is just an harmonic median of the following two proportions:

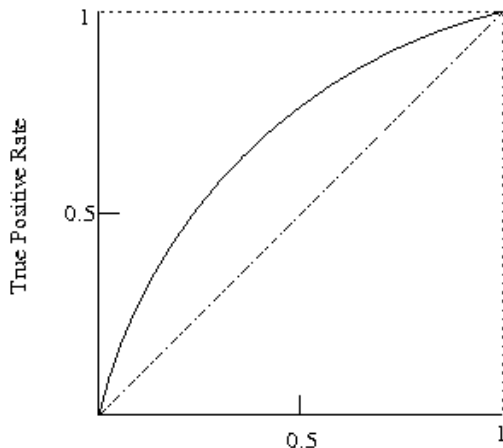
$$f1 = \frac{2 \cdot \textit{precision} \cdot \textit{recall}}{\textit{precision} + \textit{recall}} \quad (11)$$

where

- Precision ($\textit{precision} = TP / TP + FP$) is the proportion of positive predictions that are correct (no. of good item recommended / no. of all recommendations)
- *recall* is the *true positive rate* defined as $TP / TP + FN$, no. of good items recommended / no. of good items

Receiver Operating Characteristic (ROC) Curve

The ROC curve provides a graphical visualisation to analyse the relationship between the *true positive rate* and the *true negative rate*.



False Positive Rate

Receiver Operating Characteristic (ROC) Curve

The optimal point in the graph is the top-left corner, i.e., all positive instances are classified correctly and no negative instances are misclassified as positive. The diagonal line on the graph represents the scenario of randomly guessing the class, and so it represents the minimum baseline for all classifiers.

The Area Under the ROC Curve (AUC) also provides a quality measure with a single value and is used to compare different algorithms.

Rule Evaluation Measures

- The *Support* of a rule refers to the ratio between the number of instances satisfying both the antecedent and the consequent part of a rule and the total number of instances.

$$Sup(R_i) = \frac{n(Antecedent \cdot Consequent)}{N} = \frac{TP}{N} \quad (12)$$

where the $n(Antecedent \cdot Consequent)$ corresponds to the TP and N is the total number of instances.

- *Confidence* ($Conf$), also known as *Precision* or *Positive Predictive Value* (PPV) of a rule is the percentage of positive instances of a rule, i.e. relative frequency of the number of instances satisfying the both the condition (*Antecedent*) and the *Consequent* and the number of instances satisfying the only the condition.

$$Conf(R_i) = \frac{n(Antecedent \cdot Consequent)}{n(Antecedent)} = \frac{TP}{TP + FP} \quad (13)$$

Rule Evaluation Measures - Association Rules

- *Coverage* of a rule (Cov) is the percentage of instances covered by a rule of the induced set of rules

$$Cov(R_i) = p(Cond) = \frac{n(Cond)}{N} \quad (14)$$

where R_i is a single rule, $n(Cond)$ is the number of instances covered by condition $Cond$ and N is the total number of instances.

Numeric Evaluation

With numeric evaluation (e.g. difference between predicted rating and actual rating), the following measures are typically used.

- Mean Absolute Error

$$MAE = \frac{\sum_{(u,i)} |p_{(u,i)} - r_{u,i}|}{N} \quad (15)$$

- Root mean squared error

$$RMSE = \sqrt{\frac{\sum_{(u,i)} (p_{(u,i)} - r_{u,i})^2}{N}} \quad (16)$$

RMSE penalises more larger errors.

Other are also possible such as Mean-squared error, Relative-squared error

- 1 Introduction
- 2 Recommender Systems
- 3 Algorithms
- 4 Evaluation of Recommender Systems
- 5 Dimensionality Reduction**
- 6 Tools
- 7 Problems and Challenges
- 8 Conclusions
- 9 References

Dimensionality Reduction

An different approach to estimating the blank entries in the utility matrix is to break the utility matrix into the product of smaller matrices.

This approach is called SVD (Singular Value Decomposition) or a more concrete approach based on SVD called UV-decomposition.

Singular Value Decomposition

Singular Value Decomposition is a way of breaking up (factoring) a matrix into three simpler matrices.

$$M = U_{m,m} \times S_{m,n} \times V_{n,n}^T$$

where S a diagonal matrix with non-negative sorted values

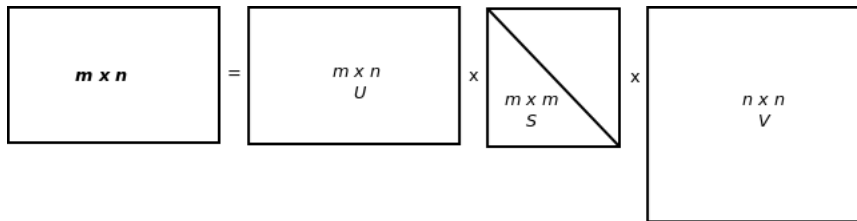
Finding new recommendations:

- 1 New user B comes in with some ratings in the original feature space $[0,0,2,0,4,1]$
- 2 Map B to a k dimensional vector: $BUS^1 = [-0.25, -0.15]$
- 3 Any distance measure can be used in the reduced space to provide recommendations

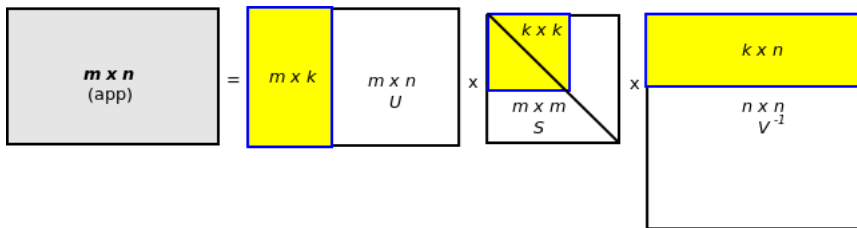
Example taken from:

<http://www.ischool.berkeley.edu/node/22627>

Singular Value Decomposition



Singular Value Decomposition



Singular Value Decomposition

Example using R

```
N=matrix(c(5,2,1,1,4,  
           0,1,1,0,3,  
           3,4,1,0,2,  
           4,0,0,0,3,  
           3,0,2,5,4,  
           2,5,1,5,1),byrow=T, ncol=5)
```

```
svd<-svd(N)
```

```
S <- diag(svd$d)
```

```
svd$u %*% S %*% t(svd$v)
```

```
svd <- svd(N,nu=3,nv=3)
```

```
S <- diag(svd$d[1:3])
```

```
svd$u %*% S %*% t(svd$v)
```

Singular Value Decomposition

Selecting $k=3$:

```
svd <- svd(N, nu=3, nv=3)
S <- diag(svd$d[1:3])
svd$u %*% S %*% t(svd$v)
```

```
> svd$u %*% S %*% t(svd$v)
```

	[,1]	[,2]	[,3]	[,4]	[,5]
[1,]	4.747427	2.077269730	1.0835420	0.9373748	4.237575
[2,]	1.602453	0.513149695	0.3907818	0.4122864	1.507976
[3,]	3.170104	3.964360658	0.5605769	0.1145642	1.913918
[4,]	3.490307	0.136595437	0.6203060	-0.2117173	3.392280
[5,]	3.064186	-0.008805885	1.7257425	5.0637172	3.988441
[6,]	1.821143	5.042020528	1.3558031	4.8996100	1.111007

Singular Value Decomposition

New user with ratings: 0,0,2,0,4,1

```
b <-matrix(c(0,0,2,0,4,1),byrow=T, ncol=6)
S1 <- diag(1/svd$d[1:3])
b %*% svd$u %*% S1
```

```
> b %*% svd$u %*% S1
      [,1]      [,2]      [,3]
[1,] -0.2550929 -0.1523131 -0.3437509
```

UV Decomposition

$M_{m,n}$ is the utility matrix

$$M = U_{m,k} \times V_{k,n}$$

$$\begin{bmatrix} m_{1,1} & m_{1,2} & \cdots & m_{1,n} \\ m_{2,1} & m_{2,2} & \cdots & m_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ m_{m,1} & m_{m,2} & \cdots & m_{m,n} \end{bmatrix} = \begin{bmatrix} u_{1,1} & \cdots & u_{1,k} \\ u_{2,1} & \cdots & u_{2,n} \\ \vdots & \ddots & \vdots \\ u_{m,1} & \cdots & u_{m,k} \end{bmatrix} \times \begin{bmatrix} v_{1,1} & v_{1,2} & \cdots & v_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ v_{k,1} & v_{k,2} & \cdots & v_{k,n} \end{bmatrix}$$

UV Decomposition: Example

$M_{m,n}$ is the utility matrix

$$M = U_{m,k} \times V_{k,n}$$

$$\begin{bmatrix} 5 & 2 & 4 & 4 & 3 \\ 3 & 1 & 2 & 4 & 1 \\ 2 & & 3 & 1 & 4 \\ 2 & 5 & 4 & 3 & 5 \\ 4 & 4 & 5 & 4 & \end{bmatrix} = \begin{bmatrix} u_{1,1} & u_{1,2} \\ u_{2,1} & u_{2,2} \\ u_{3,1} & u_{3,2} \\ u_{4,1} & u_{4,2} \\ u_{5,1} & u_{5,2} \end{bmatrix} \times \begin{bmatrix} v_{1,1} & v_{1,2} & v_{1,3} & v_{1,4} & v_{1,5} \\ v_{2,1} & v_{2,2} & v_{2,3} & v_{2,4} & v_{2,5} \end{bmatrix}$$

- 1 Introduction
- 2 Recommender Systems
- 3 Algorithms
- 4 Evaluation of Recommender Systems
- 5 Dimensionality Reduction
- 6 Tools**
- 7 Problems and Challenges
- 8 Conclusions
- 9 References

Apache Mahout

Apache Mahout machine learning library is written in Java that is designed to be scalable, i.e. run over very large data sets. It achieves this by ensuring that most of its algorithms are parallelizable (map-reduce paradigm on Hadoop).

- The Mahout project was started by people involved in the Apache Lucene project with an active interest in machine learning and a desire for robust, well-documented, scalable implementations of common machine-learning algorithms for clustering and categorization
- The community was initially driven by the paper *Map-Reduce for Machine Learning on Multicore* but has since evolved to cover other machine-learning approaches.



<http://mahout.apache.org/>

Apache Mahout

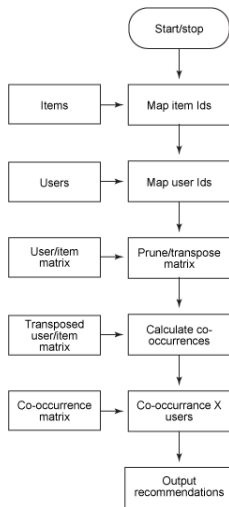
Mahout is designed mainly for the following use cases:

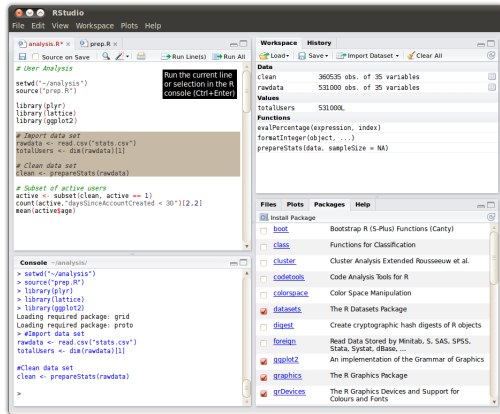
- ➊ Recommendation mining takes user's behaviour and from that tries to find items users might like.
- ➋ Clustering takes e.g. text documents and groups them into groups of topically related documents.
- ➌ Classification learns from existing categorized documents what documents of a specific category look like and is able to assign unlabelled documents to the (hopefully) correct category.
- ➍ Frequent item-set mining takes a set of item groups (terms in a query session, shopping cart content) and identifies, which individual items usually appear together.

Apache Mahout

- Mahout is built on top of Hadoop:
`http://hadoop.apache.org/`
which implements the MapReduce model.
- Hadoop is most used implementation of MapReduce model and Mahout is one of the projects of the Apache Software Foundation. Hadoop divides data into small pieces to process it and in case of failure, repeats only the pieces that failed. In order to achieve this objective, Hadoop uses his own file system called HDFS (Hadoop Distributed File System).

Apache Mahout Workflow





R with RecommenderLab and other packages for text mining.

R and extensions

RecommenderLab

- <http://lyle.smu.edu/IDA/recommenderlab/>
- <http://cran.r-project.org/web/packages/recommenderlab/>

Arules (for association rules)

- <http://lyle.smu.edu/IDA/arules/>
- <http://cran.r-project.org/web/packages/arules/>

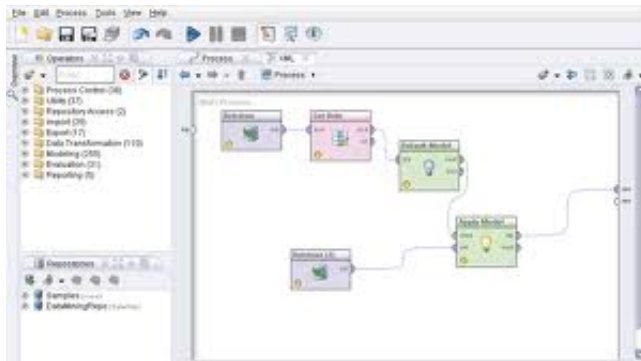
Text mining

- <http://cran.r-project.org/web/packages/tm/>

Examples and further pointers:

- <http://www.rdatamining.com/>

Rapidminer



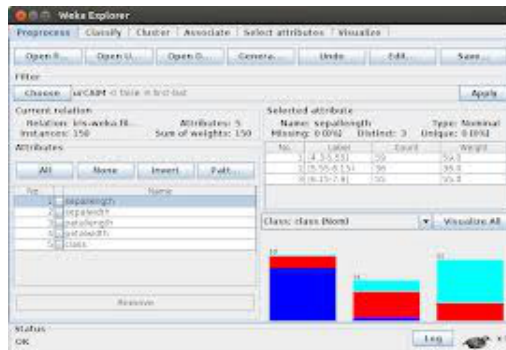
Rapidminer and its plug-in for recommendation.

KNIME



KNIME

Weka



Weka

- 1 Introduction
- 2 Recommender Systems
- 3 Algorithms
- 4 Evaluation of Recommender Systems
- 5 Dimensionality Reduction
- 6 Tools
- 7 Problems and Challenges**
- 8 Conclusions
- 9 References

Cold start Problem

When a new user or new item is added to the system, it knows nothing about them and therefore, it is difficult to draw recommendations.

Explore/Explode

Recommend items at random to a small number of randomly chosen users. To do so, we need to removed one of the highly ranked item to include the random items.

There is trade-off between recommend new items or highly ranked items:

- Exploiting a model to improve quality
- Exploring new items that can be good items and help to

- 1 Introduction
- 2 Recommender Systems
- 3 Algorithms
- 4 Evaluation of Recommender Systems
- 5 Dimensionality Reduction
- 6 Tools
- 7 Problems and Challenges
- 8 Conclusions**
- 9 References

Conclusions

Nowadays, recommendation systems are increasingly gaining notoriety due to their high number of applications.

The users cannot manage all the information available on Internet because of this is necessary some kind of filters or recommendations. Also the companies want to offer to the user specific information to increase the purchases.

The contest organized by Netflix resulted in a big jump in the research of the recommender systems, more than 40,000 teams were trying to create a good algorithm. The work of Krishnan [6] was also very interesting trying to compare the recommendation results of machine against humans. As stated in the article, the machine won in most of the comparatives because machines can handle more data than humans can.

Current and Future Work

The following improvements can be applied to the recommendation systems [9]:

- Improve the security regarding false ratings or users. E.g., Second Netflix challenge was in part cancelled because it was possible to identify users based on their votes on IMBD.
- Take the advantage of the social networks:
 - The users likes the same as his friends.
 - Explore the social graph of the users.
 - Potential friends could be suggested using recommendation techniques.
- Improve the evaluation method when there are no ratings.
- Proactive recommender systems.
- Privacy preserving recommender systems.
- Recommending a sequence of items (e.g. a play list).
- Recommender systems for mobile users.

- 1 Introduction
- 2 Recommender Systems
- 3 Algorithms
- 4 Evaluation of Recommender Systems
- 5 Dimensionality Reduction
- 6 Tools
- 7 Problems and Challenges
- 8 Conclusions
- 9 References**

References I



Deepak Agarwal.

Offline components: Collaborative filtering in cold-start situations.
Technical report, YAHOO, 2011.



Chris Anderson.

The Long Tail: Why the Future of Business Is Selling Less of More.
Hyperion, 2006.



Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth.

The kdd process for extracting useful knowledge from volumes of data.
Communications of the ACM, 39(11):27–34, November 1996.



A. Galland.

Recommender systems.
Technical report, INRIA-Saclay, 2012.

References II



Joseph A. Konstan and John Riedl.

Recommender systems: from algorithms to user experience.

User Modeling and User-Adapted Interaction, 22(1-2):101–123, April 2012.



Vinod Krishnan, Pradeep Kumar Narayanashetty, Mukesh Nathan, Richard T. Davies, and Joseph A. Konstan.

Who predicts better?: results from an online study comparing humans and an online recommender system.

In *Proceedings of the 2008 ACM conference on Recommender systems*, RecSys '08, pages 211–218, New York, NY, USA, 2008. ACM.

References III



Yashar Moshfeghi, Deepak Agarwal, Benjamin Piwowarski, and Joemon M. Jose.

Movie recommender: Semantically enriched unified relevance model for rating prediction in collaborative filtering.

In *Proceedings of the 31th European Conference on IR Research on Advances in Information Retrieval*, ECIR '09, pages 54–65, Berlin, Heidelberg, 2009. Springer-Verlag.



Anand Rajaraman, Jure Leskovec, and Jeffrey David Ullman.
Mining of Massive Datasets.

Cambridge University Press, 2011.



Francesco Ricci.

Information search and retrieval, 2012.

References IV



Francesco Ricci, Lior Rokach, and Bracha Shapira.

Recommender Systems Handbook, chapter Introduction to Recommender Systems Handbook, pages 1–35.

Springer, 2011.